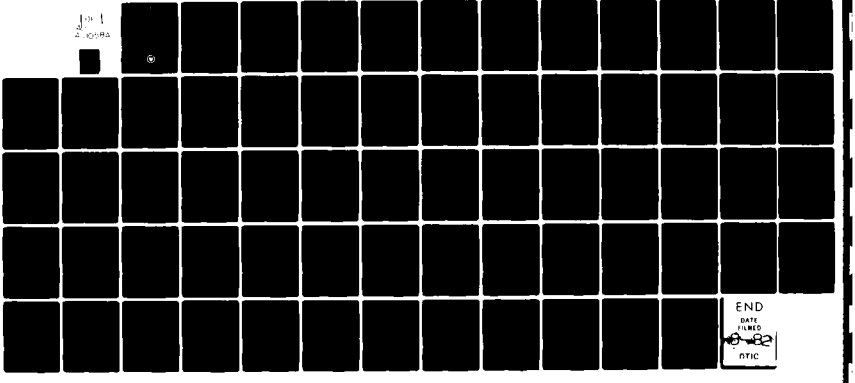
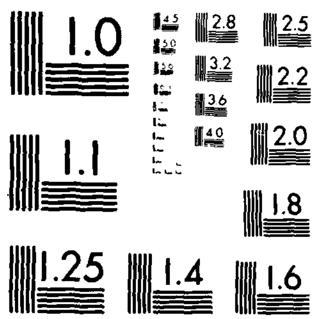
AD-A110 504 PITTSBURGH UNIV PA.INST FOR COMPUTATIONAL MATHEMATICS--ETC F/G 12/1
A PROGRAM FOR A LOCALLY-PARAMETRIZED CONTINUATION PROCESS.(U)
OCT 81 W C RHEINBOLDT, J V BURKARDT N00014-77-C-0623
UNCLASSIFIED ICMA-81-30 NL





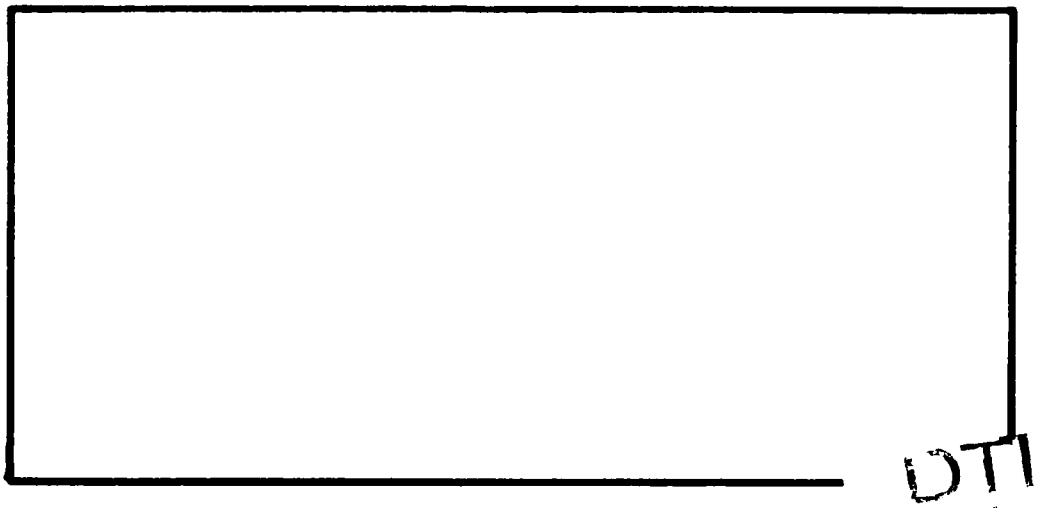
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

LEVEL II

12

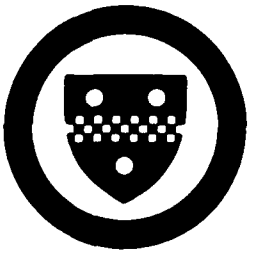
AD A110584

INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS



Department of Mathematics and Statistics

University of Pittsburgh



81 11 [illegible] 047

Technical Report ICMA-81-30 October 1981

A Program for a Locally-Parametrized Continuation Process*)

by

Werner C. Rheinboldt and John V. Burkardt

Institute for Computational Mathematics and Applications
Department of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15261
U.S.A.

*) This work was supported in part by the National Science Foundation under Grant MCS-78-05299 and the Office of Naval Research under Contract N00014-77-C-0623.

1. INTRODUCTION

The study of many equilibrium phenomena leads to non-linear equations which involve a number of intrinsic parameters. Interest then centers rarely on the determination of a few specific solutions of the equations for fixed parameter values but rather on an assessment of the behavior of these solutions under general variations of the parameters. For example, in structural analysis the parameters may characterize load points and load directions, material properties, geometrical data, etc. The set of all solutions and associated parameter values has been called the equilibrium surface of the structure (see eg. [31]). This equilibrium surface provides considerable insight into the behavior of the structure and the stability properties (see eg. [23], [32] for further discussions and various examples). From a numerical viewpoint the question then is to analyze computationally the shape and characterize particular features of this equilibrium surface.

In nonlinear mechanics the principal tools for such a computational analysis are the so-called incremental methods. These procedures were developed more or less independently in the engineering literature. But they are now also recognized to be closely related to the continuation methods used for some time in mathematics in general and in numerical analysis in particular. The literature in this area is extensive; we refer only to [21] for a discussion about the connection between incremental approaches for structural problems and continuation methods, to [8] for a historical overview of uses of continuation techniques in mathematics and to [2], [35] for some literature survey of numerical aspects of continuation methods.

Not surprisingly there are differences between the methods used in structural engineering and numerical analysis and neither is directly suited to the analysis of an equilibrium surface. In the numerical analysis liter-

ature continuation methods are usually considered only as tools for determining a specific solution y^* of a given nonlinear operator equation $Gy = 0$. For this the equation is imbedded into a one-parameter family $H(y,t) = 0$ which has a solution $y = y(t)$ for each fixed t in some interval, say, $0 \leq t \leq 1$. (See eg. [15] for a survey of such imbeddings.) If $y(t)$ depends continuously on t and satisfies $y(0) = y^0$ and $y(1) = y^*$, where y^0 is a known point, then the numerical process constructs a sequence of points in the proximity of the path $y(t)$, $0 \leq t \leq 1$, starting at y^0 and ending at the desired point y^*. On the other hand, in structural mechanics incremental methods usually are designed to follow numerically a specific load curve parametrized by a load intensity. Hence, while in the imbedding approach the parameter is essentially artificial, in the incremental procedures it has an intrinsic meaning for the application, and, even more importantly, there is no longer a fixed endpoint which is the aim of the computation, but the load curve itself is of interest.

For a numerical analysis of a given equilibrium surface we need to consider continuation-methods in a broader sense as a collection of numerical procedures for completing at least the following three basic tasks:

(1.1)

(i) Follow numerically any curve on the surface specified by a particular combination of parameter values with one degree of freedom.

(ii) On any such curve determine the exact location of target points where a given state variable has a specified value.

(iii) On such a curve identify and compute exactly the critical points where stability may be lost.

Beyond this various more special tasks may arise as, for example, the following ones:

(iv) From any one of the critical points determined under (iii) follow a path in the critical boundary.

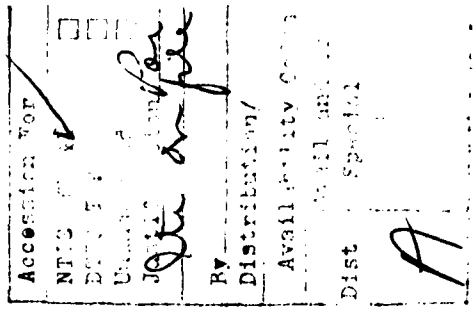
(1.2)

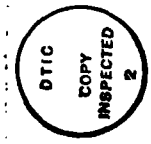
(v) On any one of the curves (i) determine the location of bifurcation points and the paths intersecting at that point.

Methods which are either directly applicable or can be readily adapted to completing these various tasks have been proposed by various authors. In particular, for (i) the literature is very large and we refer here only to the mentioned surveys [2], [35]. Methods relating to (iii) were described, for instance, in [1], [20], [22], [33], [34], and for (iv) and (v) we refer to [28] and [10], [25], respectively, where also further references are given.

So far only a few library programs for performing these various tasks have been published. Without claim for completeness we mention here [14], [38]. Each one of these programs has the objective of computing a specified solution curve of a nonlinear equation by a continuation approach along the lines sketched above. In this paper, we present a new library package specifically written with the objective of completing the three basic tasks (1.1) (i), (ii), (iii). The package can be expanded to incorporate facilities for (1.2) (iv), (v), but this will not be addressed here. The package is based on the continuation approaches introduced in [26], [27] and incorporates some of the concepts of steplength determination discussed in [7]. At the same time, new techniques of parameter adaptation are utilized here based on a prediction of changes in the curvature of the continuation path.

As with all programming packages further improvements are possible. For example, it is planned to introduce an automatic first step selection and a function-scaling option. Special versions incorporating facilities for the tasks (1.2) are also being designed. But since all these changes are built on the present package the presentation of a documentation of PITCON in its basic form appeared desirable and justified.





2. BASIC FORMULATION

Generally, after suitable discretizations, the equilibrium problems mentioned in the introduction lead to a finite-dimensional, non-linear equation of the form

$$G(y,p) = 0 \tag{2.1}$$

where $y \in R^m$ is a vector of state variables, $p \in R^r$ a vector of parameters, and $G: R^m \times R^r \to R^m$ a given function. Then we are interested in the features of the set

$$E(G) = \{(y,p) \in R^m \times R^r;\ G(y,p) = 0\} \tag{2.2}$$

of all solutions of (2.1). Under well-known conditions $E(G)$ represents an r-dimensional manifold in $R^m \times R^r$.

In most applications, interest centers on tracing paths on $E(G)$ which are characterized by $r-1$ relations between the parameters. In other words, we are given a suitable mapping $K: R^r \to R^{r-1}$ and wish to compute the subset of $E(G)$ defined by the augmented equations

$$\begin{aligned} G(y,p) &= 0, \\ Kp &= 0. \end{aligned} \tag{2.3}$$

In this formulation we should include the parameters in the list of variables, in which case,(2.3) represents a system with one more variable then equations. Then, for ease of notation, it is reasonable to combine the vectors y and p into one vector x of dimension $n = m+r$. Moreover, from the viewpoint of our package of library programs it is natural to assume that both mappings

G and K of (2.3) are provided for by the user. In other words, we may write (2.3) as one equation

$$Fx = 0 \tag{2.4}$$

with a user-specified mapping $F: R^n \to R^{n-1}$. Note, however, that in this underdetermined equation (2.4) no one variable is explicitly identified as continuation variable as is typical in the incremental and continuation methods mentioned in the introduction.

We assume here that the given mapping F has the following properties:

(2.5)
(i) F is continuously differentiable on R^n.
(ii) The derivative $DF(x)$ of F is locally lipschitzian on R^n.
(iii) The regularity set $R(F) = \{x \in R^n;\ \text{rank } DF(x) = n-1\}$ is non-empty and therefore an open subset of R^n.

From (2.5) it follows (see [26]) that the tangent map specified by

$$T: R(F) \to R^n, \quad DF(x)Tx = 0, \quad ||Tx||_2 = 1, \quad \det\begin{pmatrix} DF(x) \\ (Tx)^T \end{pmatrix} > 0 \tag{2.6}$$

is uniquely determined and locally lipschitzian on $R(F)$. Furthermore, (2.5) implies that the regular solution set $E(F) \cap R(F)$ of F is either empty or a one-dimensional C^1-manifold in the open set $R(F)$. Our objective is to determine numerically a non-empty connected component E^* of $E(F) \cap R(F)$. It is well-known (see eg. [17]) that such a component E^* is diffeomorphic either to the circle or to some interval (that is, some connected subset) of R^1. Hence, E^* is uniquely determined by any one of its points $x^0 \in E(F) \cap R(F)$

and we denote this by writing $E^*(F,x^0)$., Note that for any $x^1 \in E^*(F,x^0)$ we have $E^*(F,x^1) = E^*(F,x^0)$.

A parametrization by arclength of $E^*(F,x^0)$ is a solution of the initial value problem

$$\dot{x} = Tx, \quad x(0) = x^0. \tag{2.7}$$

Note that, since T is locally Lipschitzian, (2.7) has a unique solution which cannot terminate inside $R(F)$. Evidently standard ODE-solvers may be applied to solve (2.7) numerically. This has been pursued for some time in the literature (see eg. [3], [6], [13], [37]). Independent of this, the choice of the arclength for the parametrization of $E^*(F,x^0)$ has been proposed by many authors. Notably H. B. Keller and his co-workers (see eg. [10], [11]) have advocated this choice for some time. It is also the basis of incremental procedures given in [5], [29] and has been more or less implicit in various papers in the field.

Our programs here are based more generally on the structure of $E^*(F,x^0)$ as a one-dimensional manifold and use a local parametrization at each point computed along $E^*(F,x^0)$. A natural class of such local parameters are the n components of the vector x. We call a process based on this choice of parametrization a locally-parametrized continuation method.

3. OUTLINE OF THE PROCESS AND BASIC STEPS

As noted before, our objective is to determine numerically a non-empty component $E^*(F,x^0)$ of the regular solution set $E(F) \cap R(F)$. For the discussion it is useful to consider a parametrization by arclength of $E^*(F,x^0)$, that is, a function $x: J \to E^*(F,x^0)$ which maps some interval $J \subset R^1$ diffeomorphically onto some open subset of $E^*(F,x^0)$ such that $||\dot{x}(s)||_2 = 1$ for $s \in J$. We may assume also that $x(0) = x^0$, $0 \in J$.

The process described here belongs to the class of predictor-corrector continuation methods. Starting from x^0 it produces a sequence of approximations $x^k \doteq x(s_k)$, $k = 0,1,\ldots,$ corresponding to some sequence $0 = s_0 < s_1 < s_2 < \ldots$ of arclength values. Note, however, that in general the values $s_1,s_2,\ldots$ are only approximately computable and are of limited interest in most applications.

In our program the principal steps performed during one continuation step are as follows:

(3.1)

1. Initialization.
2. Check for and computation of target point, if desired.
3. Calculation of tangent vector and determination of new local continuation parameter.
4. Check for and computation of limit point, if desired.
5. Steplength computation.
6. Computation of predicted point and corrector iteration.
7. Storage of data and return.

The sequencing of these steps is dictated by the data-flow. For the description of the details it will be advantageous not to adhere to this

sequence. Instead, in the remainder of this section, we discuss the basic steps 3. and 6. Then the next section introduces the new steplength computation used in step 5. and section 5 covers steps 2. and 4. The data-handling steps 1. and 7. should be self-explanatory from the documentation of the program itself.

Let $e^1,\ldots,e^n$ be the natural basis vectors of R^n. Then it is readily verified that (see eg. [26])

$$\det\begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix} = [(e^i)^T Tx] \det\begin{pmatrix} DF(x) \\ (Tx)^T \end{pmatrix}, \quad \forall\, x \in R(F), \quad i = 1,\ldots,n, \tag{3.2}$$

where the matrix occurring on the right is non-singular. Hence, for any index i, $1 \leq i \leq n$, such that $(e^i)^T Tx \neq 0$, the solution $v \in R^n$ of the linear system

$$\begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix} v = e^n \tag{3.3}$$

is uniquely defined. Evidently, then

$$Tx = \sigma \frac{v}{||v||_2}, \tag{3.4}$$

and, in line with (2.6), we should set

$$\sigma = \operatorname{sign}(v^T e^i) \operatorname{sign} \det\begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix}. \tag{3.5}$$

As long as the solution path remains completely in $R(F)$ this is satisfactory.

But frequently in applications we may encounter a bifurcation point $x^* \notin R(F)$ where several solution paths terminate. For example, using arclength representations we may find solutions $x^j: J_j \subset R^1 \to R(F)$, $j = 1,\ldots,4$, for which $x^j(s)$ tends to x^* when s tends to one of the endpoints of J_j. Moreover, it often happens that there are pairs of these solutions, say, x^1 and x^2 for which $\lim \dot{x}^1(s) = -\lim \dot{x}^2(s)$ at x^*, (see Fig. 1). In other words, if we disregard the direction of the solutions, they appear to form one smooth curve through x^*. In such a case, when the process moves along x^1 toward x^* it usually "jumps" over x^* onto x^2. Then, unless we reverse the sign of σ in (3.4) the tangent will again point toward x^* and the process reverses direction.

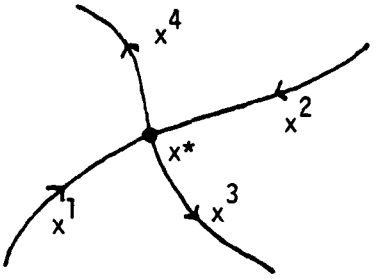


Figure 1

In order to avoid this problem suppose that the point x in (3.4) is the k-th approximation computed along the curve. Then σ is determined as follows

$$\sigma = \begin{cases} \text{dir}, & \text{if } k = 0 \\ +1, & \text{if } \operatorname{sign} v^T e^i = \operatorname{sign}(Tx^{k-1})^T e^i \\ -1, & \text{otherwise} \end{cases} \tag{3.6}$$

where dir is a user specified direction at the starting point. By comparing this value of σ with that of (3.5) we can detect if the process did jump over a bifurcation point of odd multiplicity. Obviously, bifurcation points of even multiplicity cannot be found this way.

Once the tangent Tx^k has been obtained we determine the indices j_1 and j_2 of the largest and second largest component of Tx^k in modulus,

respectively. The relation (3.2) certainly suggests that the index i_k, $1 \le i_k \le n$, of the new local continuation variable be set equal to j_1. However, if we are approaching a limit point in the j_1-th variable then this choice may be disadvantageous. Accordingly, if the following three conditions are simultaneously satisfied

$$\text{(i)} \quad |(e^{j_1})^T Tx^k| < |(e^{j_1})^T Tx^{k-1}|,$$

$$\text{(ii)} \quad |(e^{j_2})^T Tx^k| > |(e^{j_2})^T Tx^{k-1}|, \tag{3.7}$$

$$\text{(iii)} \quad |(e^{j_2})^T Tx^k| \ge \mu |(e^{j_1})^T Tx^k|,$$

with a fixed μ, $0 < \mu < 1$, then we set $i_k = j_2$. Of course, if we don't have a previous tangent vector this check has to be bypassed. The new continuation index i_k will be used for the computation of the next point x^{k+1} and its tangent Tx^{k+1}. For the tangent computation at x^0 a continuation index is assumed to be given by the user.

With the tangent Tx^k and the steplength $h_k > 0$ determined by the steplength algorithm of section 4 we compute now the predicted point $\hat{x}^k = x^k + h_k Tx^k$. Then any appropriate iterative method for the solution of the augmented equation

$$\hat{F}x \equiv \begin{pmatrix} Fx \\ (e^{i_k})^T (x - \hat{x}^k) \end{pmatrix} = 0 \tag{3.8}$$

starting from $\hat{x}^k$ may be used as a corrector process. In the program we use either the regular Newton method or its modified form in which the Jacobian at the starting point is held fixed.

Let $y^0 = \hat{x}^k, y^1, y^2, \ldots$ be the iterates produced in this way. The process has to incorporate provisions for monitoring the convergence and for aborting the iteration as soon as divergence is suspected. In the program non-convergence is declared if any one of the following three conditions is true

$$\text{(i)} \quad ||\hat{F}y^j|| \geq \theta ||\hat{F}y^{j-1}|| \quad \text{for some } j \geq 1,$$

$$\text{(3.9)} \quad \text{(ii)} \quad ||y^j - y^{j-1}|| \geq \theta ||y^{j-1} - y^{j-2}|| \quad \text{for some } j \geq 2,$$

$$\text{(iii)} \quad j \geq j_{max}.$$

For the constant θ we use $\theta = 1.05$ except in the first check of (3.9) (i) where $\theta = 2$ is chosen. The maximal iteration count j_{max} depends on the method. For the regular Newton process we set $j_{max} = 10$ and double this for the modified method. In the case of non-convergence the predictor step is reduced by a given factor, for example 1/3, unless the resulting step is below a given minimal steplength.

Convergence is declared if either one of the two conditions holds for an iterate:

$$\text{(i)} \quad ||\hat{F}y^j|| \leq 8\, \varepsilon_{mach} \quad \text{for some } j \geq 0,$$

$$\text{(3.10)}$$

$$\text{(ii)} \quad (||Fy^j|| \leq \varepsilon_{abs}) \quad \text{and} \quad (||y^j - y^{j-1}|| \leq \varepsilon_{abs} + \varepsilon_{rel}||y^j||) \quad \text{for some } j \geq 1.$$

The tolerances ε_{abs}, ε_{rel} are user specified and ε_{mach} is the smallest floating point number such that $1. = 1. + \varepsilon_{mach}$. In both tests (3.9) and (3.10) the maximum norm is used.

4. THE STEPLENGTH ALGORITHM

For the points x^k, $k = 0,1,\ldots$, approximating the continuation curve $x: J \to E^*(F,x^0)$ the achievable error $||x^k - x(s_k)||$ is solely determined by the termination criterion (3.10) of the corrector process. In contrast to this the standard ODE-solvers involve a corrector equation obtained by extrapolation for which the solutions are not, in general, on the exact curve. As a consequence the available error for the ODE-solvers depends on the history of the process up to that point, and this in turn has a strong influence on the step-selection. On the other hand, for our continuation process any step $h_k > 0$ along the Euler line is acceptable in principle if only the corrector converges from the predicted point $\hat{x}^k$. Moreover, in [26] it was shown that any compact segment of the continuation curve in $R(F)$ has an ε-neighborhood for some $\varepsilon > 0$ in which Newton's method will converge to the curve.

This suggests that we estimate the radius of convergence of the corrector process at the computed points and extrapolate these radii to the next point about to be determined. In practice the estimate of a convergence radius at some continuation point would have to be based on the corrector iterates which led to that point. Unfortunately, as was proved in [7], this represents insufficient information for obtaining such an estimate. On the other hand, an approach was presented in [7] which allows for an assessment of the convergence quality of the particular sequence of corrector iterates.

For details of this approach we refer to the cited article. In brief, let $\{y^i\}$ be a given sequence with limit y^* generated by an iterative process and denote the errors by $e_i = ||y^i - y^*||$, $i = 0,1,\ldots$. The definition of any convergence measure is based on a hypothetical model of the behavior of the errors. For example, if $\{y^i\}$ converges linearly it is reasonable to assume that

(4.1) $$0 \leq e_{i+1} \leq \lambda e_i, \quad i = 0,1,\ldots$$

with some constant λ, $0 < \lambda < 1$, depending on $\{y^i\}$. Suppose now that the process was terminated with the iterate y^{i*}. Then

(4.2) $$\tilde{\lambda} = \tilde{\omega}^{1/(i*-1)}, \quad \tilde{\omega} = \frac{||y^{i*}-y^{i*-1}||}{||y^{i*}-y^{0}||}, \quad i* \geq 2,$$

represents a computable estimate of λ.

In the setting of our continuation process suppose now that the y^i, $i = 0,1,\ldots,$ are the corrector iterates leading from the current predicted point $\hat{x}^k = y^0$ to the new continuation point $x^{k+1} = y^{i*}$. Then

(4.3) $$\delta_k = ||\hat{x}^k - x^{k+1}|| = ||y^0 - y^{i*}||$$

is the correction-distance. For the modified Newton method the convergence is indeed linear, and a reasonable aim in the construction of the steps along the curve is to ensure that the number of corrector iterates remains about constant. In other words, we aim at taking always, say, $m*$ corrector steps. Hence, under the heuristic assumption that the error model (4.1) remains valid for some interval of starting errors e_0 around δ_k, we should have begun with an "ideal starting error" $\delta_k^* = \theta_k \cdot \delta_k$ such that

(4.4) $$\tilde{\lambda}^{m*}\, \delta_k^* \doteq \tilde{\lambda}^{i*}\, \delta_k$$

and therefore

(4.5) $$\theta_k = \tilde{\lambda}^{i^*-m^*} \equiv \tilde{\omega}^{(i^*-m^*)/(i^*-1)}.$$

In our program we use $m^* = 10$ for the modified Newton method and enforce always that $0.125 \leq \theta_k \leq 8$.

This technique is also readily applicable for Newton's method. In [7] two different hypothetical error models for the Newton process were discussed. Here we use only one of these models, namely the one arising in the attraction theorem formulated in [24]. In essence, under certain conditions about the equation and the desired limit y^* of the Newton process there exists a radius $r^* > 0$ such that for any starting point y^0 in the ball $B(y^*,r^*)$ the relative errors $\varepsilon_i = e_i/r^*$, $i = 0,1,\ldots,$ satisfy

(4.6) $$0 \leq \varepsilon_{i+1} \leq \phi(\varepsilon_i), \quad i = 0,1,\ldots, \quad \phi(t) = \frac{t^2}{3-2t}, \quad 0 \leq t \leq 1.$$

The radius r^* depends on global information about the equation and is not accessible. If $0 < \varepsilon_0 < 1$ and the $\{\varepsilon_i\}$ satisfy (4.6) then we have

(4.7) $$\varepsilon_i \leq \eta_i \equiv \phi^i(\eta_0) \equiv \frac{3}{1 + 2\cosh 2^i\alpha}, \quad i = 0,1,\ldots, \quad \eta_0 = \varepsilon_0,$$

where α is the unique positive solution of

(4.8) $$\psi(\alpha) = \eta_0, \quad \psi(\alpha) = \frac{3}{1 + 2\cosh \alpha}.$$

Moreover, for any ω, $0 < \omega < 1$, and $i^* \geq 2$ the equation

$$(4.9) \qquad \frac{1}{\psi(\alpha)}\,\phi^{i^*-1}(\psi(\alpha)) \equiv \frac{1 + 2\cosh\alpha}{1 + 2\cosh 2^{i^*-1}\alpha} = \omega$$

has a unique solution $\alpha > 0$.

Now suppose that $\{y^i\}$ denotes the sequence of Newton iterates and that the process was terminated at y^{i^*}. As in the linear case we use the approximation

$$(4.10) \qquad \tilde{\omega} = \frac{||y^{i^*}-y^{i^*-1}||}{||y^{i^*}-y^0||} \doteq \frac{e_{i^*-1}}{e_0} = \frac{\varepsilon_{i^*-1}}{\varepsilon_0} \leq \frac{\eta_{i^*-1}}{\eta_0}$$

and compute with this $\tilde{\omega}$ the solution $\tilde{\alpha}$ of (4.9) which gives the estimate $\tilde{\eta}_0 = \psi(\tilde{\alpha})$ of ε_0. Now we proceed as before and obtain the factor

$$(4.11) \qquad \theta_k^* = \frac{\delta_k^*}{\delta_k} = \frac{\tilde{\eta}_0'}{\tilde{\eta}_0}$$

for the ideal starting error by determining the unique solution $\tilde{\eta}_0'$, $0 < \tilde{\eta}_0' < 1$, of

$$(4.12) \qquad \phi^{m^*}(\tilde{\eta}_0') = \phi^{i^*}(\tilde{\eta}_0).$$

Since the iterates ϕ^i are explicitly known the various equations are not difficult to solve numerically. However, for the computation it is more advantageous to introduce a least squares fit of θ_k as a function of $\tilde{\omega}$ for all relevant values of i^*. In the program we use $m^* = 4$ and the approximations for θ_k given in Table 1. Note that as before we restrict θ_k to the interval $0.125 \leq \theta_k \leq 8$.

i^*	$\tilde{\omega} \in [a,b]$		θ_k
	a	b	
2	0.8735115	1	1
	0.1531947	0.8735115	0.9043128 - 0.7075675 ln ω
	0.03191815	0.1531947	-4.667383 - 3.677482 ln ω
	0	0.03191815	8
3	0.4677788	1	1
	0.6970123(-3)	0.4677788	0.8516099 - 0.1953119 ln ω
	0.1980863(-5)	0.6970123(-3)	-4.830636 - 0.9770528 ln ω
	0	0.1980863(-5)	8
4	0	1	1
5	0.3339946(-10)	1	1.040061 + 0.03793395 ln ω
	0	0.3339946(-10)	0.125
6	0.1122789(-8)	1	1.042177 + 0.04450706 ln ω
	0	0.1122789(-8)	0.125
≥ 7	0	1	0.125

Table 1

We turn now to the algorithm for the determination of the steplength $h_k > 0$ along the Euler line $\pi(t) = x^k + t\,Tx^k$ used for the prediction. In order to estimate the distance between $\pi(t)$ and the exact curve $x = x(s)$ we introduce the quadratic Hermite-Birkhoff interpolation polynomial

$$(4.13)\qquad q(t) = x^k + t\,Tx^k + \frac{1}{2}t^2 w^k,\ w^k = \frac{1}{\Delta s_k}(Tx^k - Tx^{k-1}),\ \Delta s_k = ||x^k - x^{k-1}||_2$$

for which

$$(4.14)\qquad q(0) = x^k,\ q'(0) = Tx^k,\ q'(-\Delta s_k) = Tx^{k-1}.$$

Since

$$(4.15)\qquad w^k \doteq \int_0^1 x''(s_k - \sigma\Delta s_k)d\sigma = x''(s_k - \tilde{\sigma}\Delta s_k),\quad 0 < \tilde{\sigma} < 1,$$

the quantity

$$(4.16)\qquad ||w^k||_2 = \frac{2}{\Delta s_k}\left|\sin\frac{1}{2}\alpha_k\right|,\quad \alpha_k = \arccos\left((Tx^k)^T Tx^{k-1}\right)$$

represents an approximation of the curvature of the exact point at some point between $x(s_{k-1})$ and $x(s_k)$.

It is tempting to derive from q a prediction of the curvature to be expected during the next continuation step. However, a closer computation shows that the value of the curvature of q assumes its maximum $||w^k||_2/\cos^2\frac{1}{2}\alpha_k$ at $t = -\frac{1}{2}\Delta s_k$ and that for increasing t this value decreases rapidly. For example, at $t = 0$ the curvature of q equals only $||w^k||_2\,|\cos\frac{1}{2}\alpha_k|$ and for positive t no reasonable predictive information can be gained this way.

The relation (4.15) suggests the use of the simple linear extrapolation

$$(4.17a)\qquad \gamma_k^{tent} = ||w^k||_2 + \frac{\Delta s_k}{\Delta s_k + \Delta s_{k-1}}\left(||w^k||_2 - ||w^{k-1}||_2\right)$$

for a prediction of the curvature during the next continuation step. However, this value may become negative and accordingly we use instead

$$\gamma_k = \max\,(\gamma_{min}, \gamma_k^{tent}) \tag{4.17b}$$

with a given small $\gamma_{min} > 0$.

Most of the data discussed so far are sketched in Figure 2. In order to derive a formula for the desired predictor step h_k we note that

$$||q(t) - \pi(t)||_2 = \frac{1}{2}\, t^2 ||w^k||_2 \tag{4.18}$$

represents an estimate of the distance between the Euler line and the exact curve. In fact, for smooth curves the error of this estimate is asymptotically or order three in $\max(|t|, \Delta s_k)$ as this quantity tends to zero. Hence, if we want this distance to be at most equal to a tolerance $\varepsilon > 0$ then we should choose the next step as

$$t = \sqrt{\frac{2\varepsilon}{||w^k||_2}}\,. \tag{4.19}$$

It is natural to replace the curvature $||w^k||_2$ by the predicted value γ_k of (4.17) and to relate the tolerance ε to the "ideal starting error" δ_k^* obtained earlier. As Figure 2 indicates it is unreasonable to expect $\varepsilon > \Delta s_k$. Hence, we use instead

$$\varepsilon_k = \begin{cases} \varepsilon_{min}\, \Delta s_k & \text{if} \quad \delta_k^* \le \varepsilon_{min}\, \Delta s_k \\ \Delta s_k & \text{if} \quad \delta_k^* \ge \Delta s_k \\ \delta_k^* & \text{otherwise} \end{cases} \tag{4.20}$$

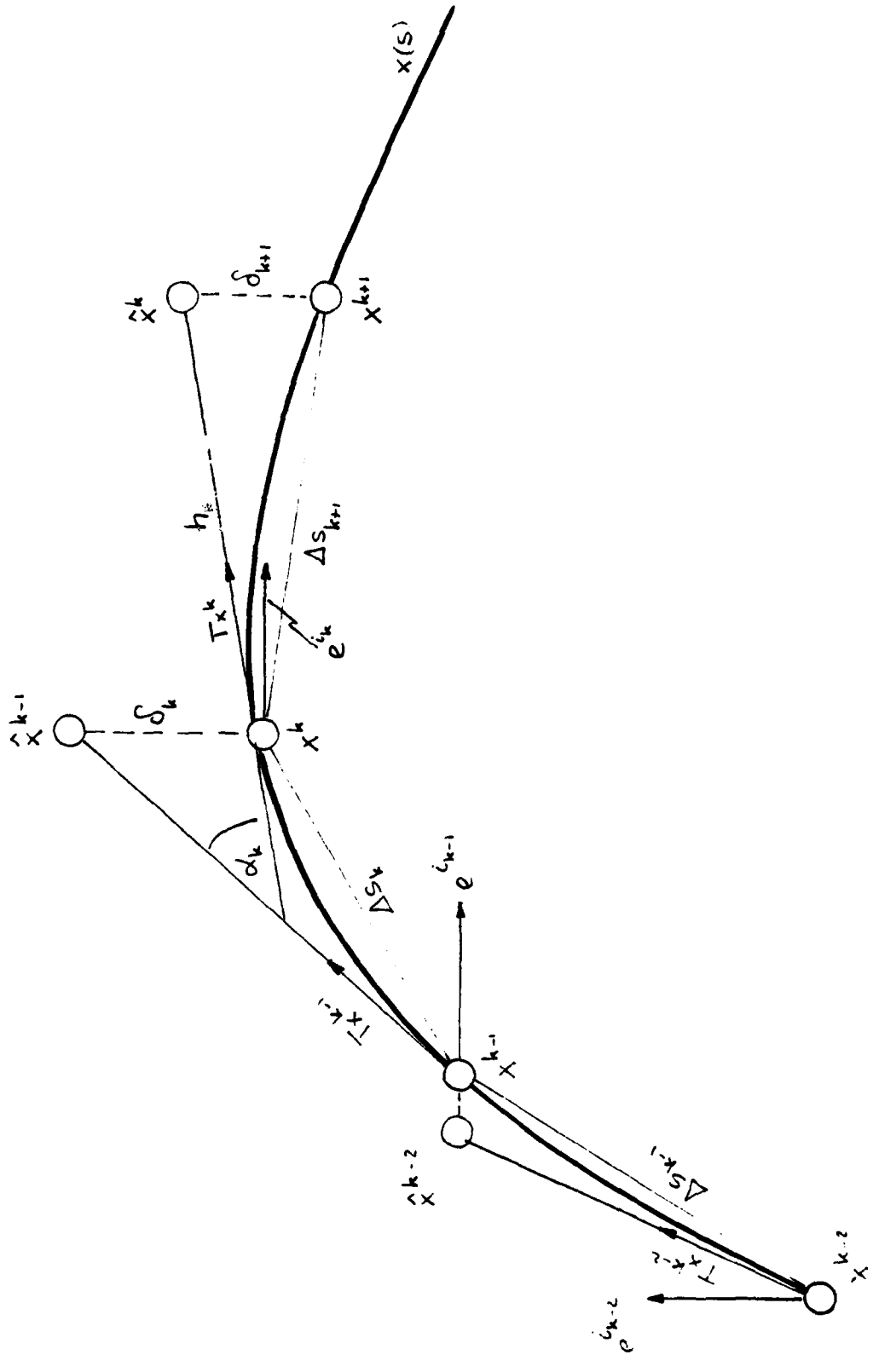

x(s)
$\hat{x}^k$
δ_{k+1}
x^{k+1}
h_k
Δs_{k+1}
T_x^k
e^{i_k}
$\hat{x}^{k-1}$
δ_k
x^k
α_k
Δs_k
$e^{i_{k-1}}$
$T_{x^{k-1}}$
x^{k-1}
$\hat{x}^{k-2}$
Δs_{k-1}
$T_{x^{k-2}}$
x^{k-2}
$e^{i_{k-2}}$

Figure 2

with a small $\varepsilon_{min} > 0$, e.g., $\varepsilon_{min} = 0.01$. Then a tentative predicted step is given by

$$h_k^{(1)} = \sqrt{\frac{2\varepsilon_k}{\gamma_k}} . \tag{4.21}$$

From the form (3.8) of the augmented equation we see that the corrector iterates remain in a hyperplane perpendicular to the basis vector e^{i_k} through the predicted point. Then Figure 2 suggests that we adjust the predicted steplength h_k so as to ensure that h_k will be approximately equal to Δs_{k+1}. There is no need to enforce this too rigidly. It suffices to define a new tentative step by the requirement

$$(e^i)^T \pi(h_k^{(2)}) = (e^i)^T q(h_k^{(1)})$$

whence,

$$h_k^{(2)} = h_k^{(1)}[1 + \frac{h_k^{(1)}}{2\Delta s_k} (1 - \frac{(e^i)^T Tx^{k-1}}{(e^i)^T Tx^k})]. \tag{4.22}$$

This formula may involve subtractice cancellation and has to be evaluated in double precision.

The final value h_k of the steplength is now obtained from $h_k^{(2)}$ by enforcing three different bounding requirements. First of all, if the previous continuation step from x^{k-1} to x^k was obtained only after a failure of the correction process and a corresponding reduction of the predicted step, then we should not allow h_k to exceed Δs_k. Secondly, as in the ODE solvers we need to control both the relative growth and the absolute size of the pre-

dictor step. Thus, we require that

$$\frac{1}{\kappa}\,\Delta s_k \le h_k \le \kappa \Delta s_k, \quad h_{min} \le h_k \le h_{max} \tag{4.23}$$

where κ is some factor, say, $\kappa = 3$, and h_{min}, h_{max} depend on the machine as well as the requirements of the problem. It should be obvious how the final step h_k is obtained from $h_k^{(2)}$ on the basis of these restrictions.

5. THE COMPUTATION OF TARGET AND LIMIT POINTS

By generating a sequence of solution points on a given curve, the continuation process reveals the shape of the curve, but there are often other items of interest that need to be studied as well. Our program is designed to pause during the continuation steps in order to seek out special points that the user has requested, namely, target and limit points.

A target point $x \in E^*(F,x^0)$ is a point on the solution curve for which the component $x_i = (e^i)^T x$ with given index $i = IT$ has a prescribed value $\tilde{x}_i = XIT$. Limit or turning points with respect to a given index $i = LIM$ are points $x \in E^*(F,x^0)$ where the i-th component $(e^i)^T Tx$ is zero. More specifically, since it is computationally unreasonable to attempt to compute zeroes of even order, we are concerned only with limit points on the continuation curve where $(e^i)^T Tx(s)$ changes sign.

It might be mentioned that bifurcation points represent another interesting, special class of points. But in that case we are not only interested in the specific location of the point but also in the solution curves that branch off from it. This is exactly the task (1.2) (v) listed earlier. The corresponding procedures (loc. cit.) would add considerably to the complexity of our program, and, since their utility tends to be of a more specialized nature, it was decided not to cover task (1.2) (v) (nor (1.2) (iv)) in the present program.

As indicated before, the determination of a target or limit point represents an interruption in the normal flow of the continuation program. After at least one step has been taken, the program has available an old point x^{k-1}, a new point x^k and the tangent vector Tx^{k-1}. Normally, then we turn to the computation of Tx^k, of the new steplength, and finally of the next point x^{k+1}. But if the index IT or LIM is non-zero then these

computations are postponed for the search of a target or limit point, respectively. We discuss these cases separately:

Target points: Suppose that a non-zero value of $i = IT$ and associated value $\bar{x}_i = XIT$ have been given. If $\bar{x}_i$ lies between $(e^i)^T x^{k-1}$ and $(e^i)^T x^k$, then it is assumed that a solution point $x \in E^*(F,x^0)$ with $(e^i)^T x = \bar{x}_i$ is nearby. In this case a point

$$y(t) = (1-t)x^{k-1} + tx^k, \quad 0 \leq t \leq 1, \tag{5.1}$$

on the secant between x^{k-1} and x^k is determined such that $(e^i)^T y(t) = \bar{x}_i$. Now with the augmenting equation $(e^i)^T x = \bar{x}_i$ the corrector process is applied, and, if it terminates successfully the resulting point is taken as the desired target. Otherwise, a failure is indicated for the target routine. In either case, the routine returns and on the next call the continuation loop will pick up from where it was interrupted. Note that in effect the target routine uses the IT-th variable for the local parametrization of the curve between x^{k-1} and x^k. This may be an inferior choice of parameter for the corrector but it allows us to enforce that the resulting target point $x \in E^*(F,x^0)$ indeed satisfies $(e^i)^T x = \bar{x}_i$. Clearly, for very large continuation steps we have no guarantee that all target points will be detected or that a target computation will succeed. Thus, the utility of the target routine will depend on the maximal allowed stepsize that has been chosen.

Limit points: If the limit point index $i = LIM$ is non-zero, then a limit point determination is carried out after a target point search has been successfully or unsuccessfully completed, provided it was called for at all.

Recall that we still have as current information the vectors x^{k-1}, x^k, and Tx^{k-1}. Now the new tangent Tx^k is evaluated and if $\operatorname{sign}(e^i)^T Tx^{k-1} \neq \operatorname{sign}(e^i)^T Tx^k$ for $i = \mathrm{LIM}\ (\neq 0)$ then a limit point search is begun. For this the index $\hat{i}$ of the largest component in modulus of the secant direction $x^k - x^{k-1}$ is chosen as a local parametrization of the curve between x^{k-1} and x^k. More specifically, suppose that $x: [s_{k-1}, s_k] \to E^*(F, x^0)$ represents the segment of the curve between x^{k-1} and x^k. Then $\hat{i}$ is assumed to be the index of a local coordinate for which there exists a bijective parameter transformation $\phi: [0,1] \to [s_{k-1}, s_k]$ such that $(e^{\hat{i}})^T y(t) = (e^{\hat{i}})^T x(\phi(t))$, $0 \leq t \leq 1$, where $y(t)$ is defined by (5.1).

Hence, we may consider the function

$$g: [0,1] \to R^1, \quad g(t) = (e^i)^T Tx(\phi(t)), \quad 0 \leq t \leq 1, \tag{5.2}$$

and our problem is to determine a zero of g. Since by assumption $\operatorname{sign} g(0) \neq \operatorname{sign} g(1)$, a rootfinder of the Dekker-Brent type can be applied. For the evaluation of $g(t)$ we use the augmenting equation $(e^{\hat{i}})^T x = (e^{\hat{i}})^T y(t)$ and apply the corrector process with $y(t)$ as starting point. If it terminates successfully with some x then Tx can be evaluated and we set $g(t) = (e^i)^T Tx$. Hence, g is certainly costly to compute and we require an efficient rootfinder to speed the convergence of the limit point routine. A specially modified version of the routine given in [4] is used in our program. Clearly, as in the case of target points, we may fail to detect a limit point if the continuation steps are too large and in such a situation the rootfinder may also fail to converge. In addition, the evaluation of g may run into difficulties when the desired limit point is near a bifurcation point.

6. SOME NUMERICAL EXAMPLES

The programs described here have been used extensively with excellent success on problems from many different areas. We include here only a few numerical examples to illustrate the operation of the programs.

Example 1. In order to present some details of the performance of the programs we consider first a very small problem which was originally formulated in [9] and subsequently used as a test case by many authors. The mapping F has here the form

$$Fx = \begin{pmatrix} x_1 - x_2^3 + 5x_2^2 - 2x_2 + 34x_3 - 47 \\ \\ x_1 + x_2^3 + x_2^2 - 14x_2 + 10x_3 - 39 \end{pmatrix}, \quad \forall\, x \in R^3. \tag{6.1}$$

For the starting point $x^0 = (15,-2,0)^T$ the solution curve passes through $x^* = (5,4,1)^T$ and this point is chosen as target.

Tables 2 and 3 show runs with the full Newton method and modified Newton method, respectively as corrector process. A starting step $h_0 = 0.3$ and maximum step $h_{max} = 25.0$ were used. The performance for the two correctors is practically the same although the step-prediction exhibits certain differences due to our assessment of the corrector distance. Clearly, the use of the modified Newton process is much less expensive and hence preferable as Table 4 shows which summarizes the total number of function and Jacobian calls including those for the target calculation. Comparative performance data given in [7] for this problem involved 22 continuation steps, 15 step reductions and 128 Jacobian evaluations. The procedure discussed in [19] required 25 continuation steps but no further details were provided in the paper.

Step	Continuation point x_1	x_2	x_3	Contin. Variable	Total Correct. Steps	Comments
0	15.000	-2.00000	0.00000	x_3	—	
1	14.705	-1.9421	0.065381	x_1	2	
2	14.285	-1.7291	0.26874	x_3	3	
3	16.906	-1.2094	0.54684	x_2	2	
4	24.918	-0.59908	0.55514	x_1	3	
5	48.974	0.71803	-0.080758	x_1	3	
6	57.928	1.2846	-0.40736	x_1	4	Step red.
7	60.052	1.5709	-0.54035	x_1	4	Step red.
8	61.666	2.0010	-0.66683	x_2	2	
9	-5.1039	4.1510	1.3464	x_2	2	Target passed
				TOTAL	25	
				Computation of target	4	

Table 2

Step	Continuation point x_1	x_2	x_3	Contin. Variable	Total Correct Steps	Comments
0	15.000	-2.0000	0.00000	x_3	–	
1	14.710	-1.9421	0.065381	x_1	3	
2	14.285	-1.7291	0.26874	x_3	4	
3	16.906	-1.2094	0.54685	x_2	1	
4	24.918	-0.59906	0.55514	x_1	6	
5	48.975	0.71810	-0.080804	x_1	5	
6	57.289	1.2847	-0.40742	x_1	6	Step red.
7	60.053	1.5711	-0.54042	x_1	5	Step red.
8	61.666	2.0013	-0.66689	x_2	2	
9	-4.4239	4.1413	1.3229	x_2	1	Target passed
				TOTAL	33	
				Computation of target	8	

Table 3

	Corrector Process	
	Newton	Mod. Newton
Function calls	41	53
Jacobian calls	38	21

Table 4

It may be noted that the solution curve has two limit points each with respect to x_1 and x_3. The two step reductions are almost unavoidable here since the curve has a long straight segment followed by a very sharp bend. The target computation is relatively expensive since the last step is extremely large due to another straight curve segment.

Example 2. Maneuvering airplanes, especially at high angles of attack, sometimes undergo sudden jumps in their response to the pilot's control inputs. The problem has been discussed extensively in the literature, see, for example, [18], [30], [39]. Without going into further details we use here a simplified version of a system of five equilibrium equations involving the roll rate (x_1), pitch rate (x_2), yaw rate (x_3), (incremental) angle of attach (x_4), side slip angle (x_5), elevator angle (x_6), aileron angle (x_7), and rudder angle (x_8). More specifically, for the particular aircraft discussed in [18] these equations have the dimensionless form

$$Fx \equiv Ax + \phi(x) = 0, \quad \forall\, x \in R^8 \tag{6.1}$$

where

$$A = \begin{pmatrix} -3.933 & 0.107 & 0.126 & 0 & -9.99 & 0 & -45.83 & -7.64 \\ 0 & -0.987 & 0 & -22.95 & 0 & -28.37 & 0 & 0 \\ 0.002 & 0 & -0.235 & 0 & 5.67 & 0 & -0.921 & -6.51 \\ 1.0 & 0 & 0 & -1.0 & 0 & -0.168 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & -0.196 & 0 & -0.0071 & 0 \end{pmatrix}$$

and

$$\phi(x) = \begin{pmatrix} -0.727\, x_2x_3 + 8.39\, x_3x_4 - 684.4\, x_4x_5 + 63.5\, x_4x_7 \\ 0.949\, x_1x_3 + 0.173\, x_1x_5 \\ -0.716\, x_1x_2 - 1.578\, x_1x_4 + 1.132\, x_4x_7 \\ -\, x_1x_2 \\ x_1x_4 \end{pmatrix} .$$

Figure 3 shows some solution curves on the three-dimensional equilibrium surface in R^8. More specifically, in all cases we fixed a value of x_6 (elevator deflection) and chose the rudder deflection $x_8 = 0$. The paths $x_6 > \omega_1$, $x_8 = 0$ with $\omega_1 \approx -0.0061771$ have two limit points, for $\omega_1 > x_6 > \omega_2$, $x_8 = 0$, with $\omega_2 \approx -0.012498$ a third limit point appears, and for $\omega_2 > x_6$, $x_8 = 0$ only one limit point remains. A similar picture arises for negative roll rates.

In all cases the programs easily detected and computed the various limit points (see Table 5). But the example also shows that even with a large number of search paths it is difficult to provide a full picture of the location of the critical boundary, that is, of the curves of limit points with respect to x_1 for $x_8 = 0$ and varying x_6, x_7. In Figure 3 the corresponding branches of limit point curves are shown as dotted lines. They were obtained with a code for the

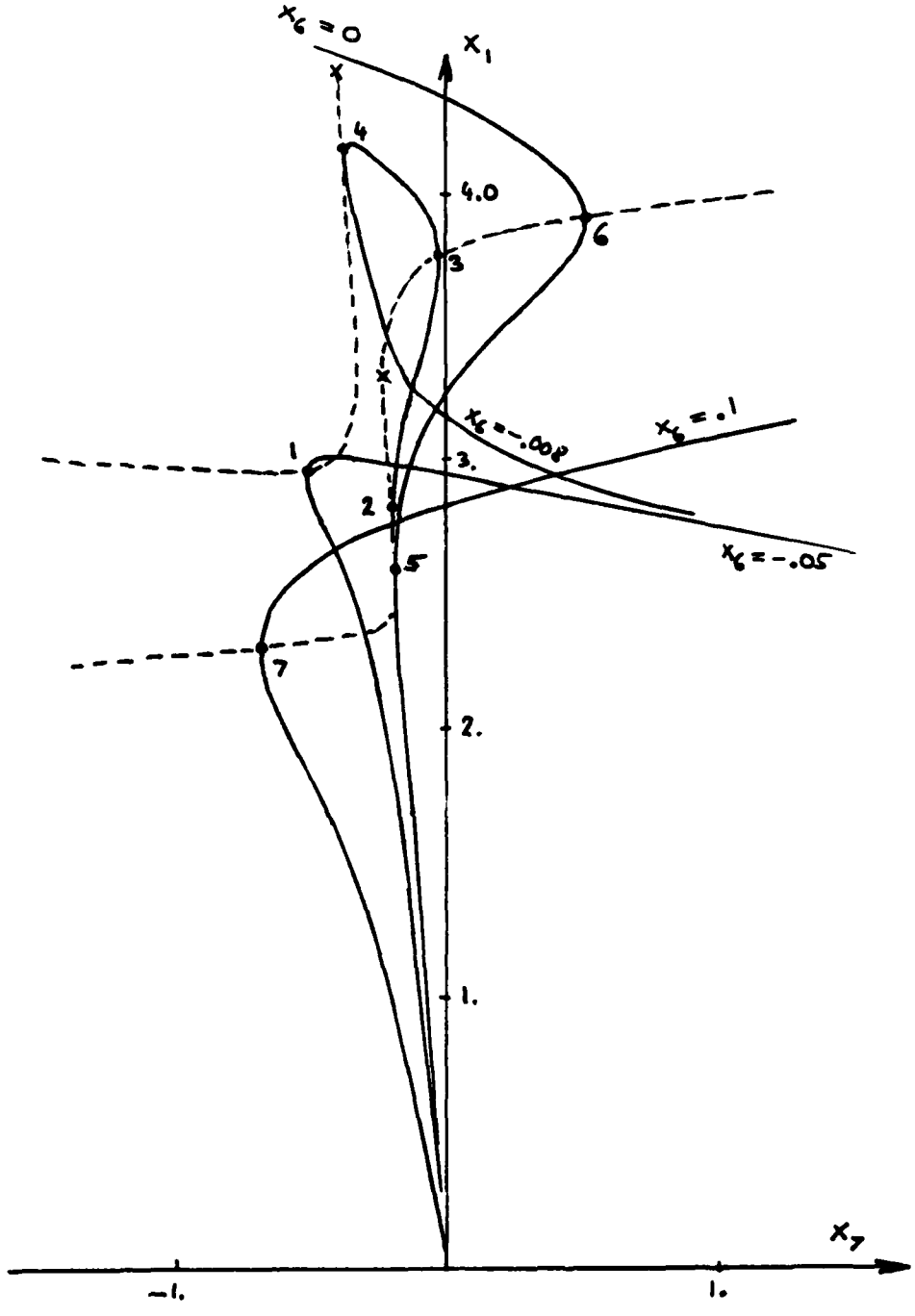


Figure 3

earlier mentioned task (1.3) (iv) (see [28]).

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	2.9649	0.82557	0.073661	0.041309	0.26735	-0.05	0.50481
2	2.8174	-0.17629	0.089926	0.026429	-0.071476	-0.008	-0.20497
3	3.7579	-0.65542	0.38658	0.092521	-0.19867	-0.008	0.006208
4	4.1638	0.089131	0.094806	0.022889	0.016232	-0.008	-0.37766
5	2.5873	-0.22355	0.054682	0.013676	-0.091687	0.0	-0.18691
6	3.9005	-1.1482	0.58156	0.13352	-0.32859	0.0	0.51016
7	2.2992	-1.4102	-0.061849	-0.079009	-0.58630	0.1	-0.68972
8	4.4565	-4.4909	1.6164	0.33091	-1.0857	0.1	10.0212

Table 5

Example 3. As an example for the numerical investigation of the equilibrium surface of a mechanical structure, we consider a clamped, thin, shallow, circular arch which has been used as a test case by various authors (see eg. [12], [16], [36]). Let U and W be the radial and axial displacements, R the arch radius, A the cross-sectional area, H the thickness, and E Young's modulus. With the dimensional displacements $u = U/H$, $w = W/H$, the total potential energy -- non-dimensionalized by dividing by $EAR(H/R)^2$ --

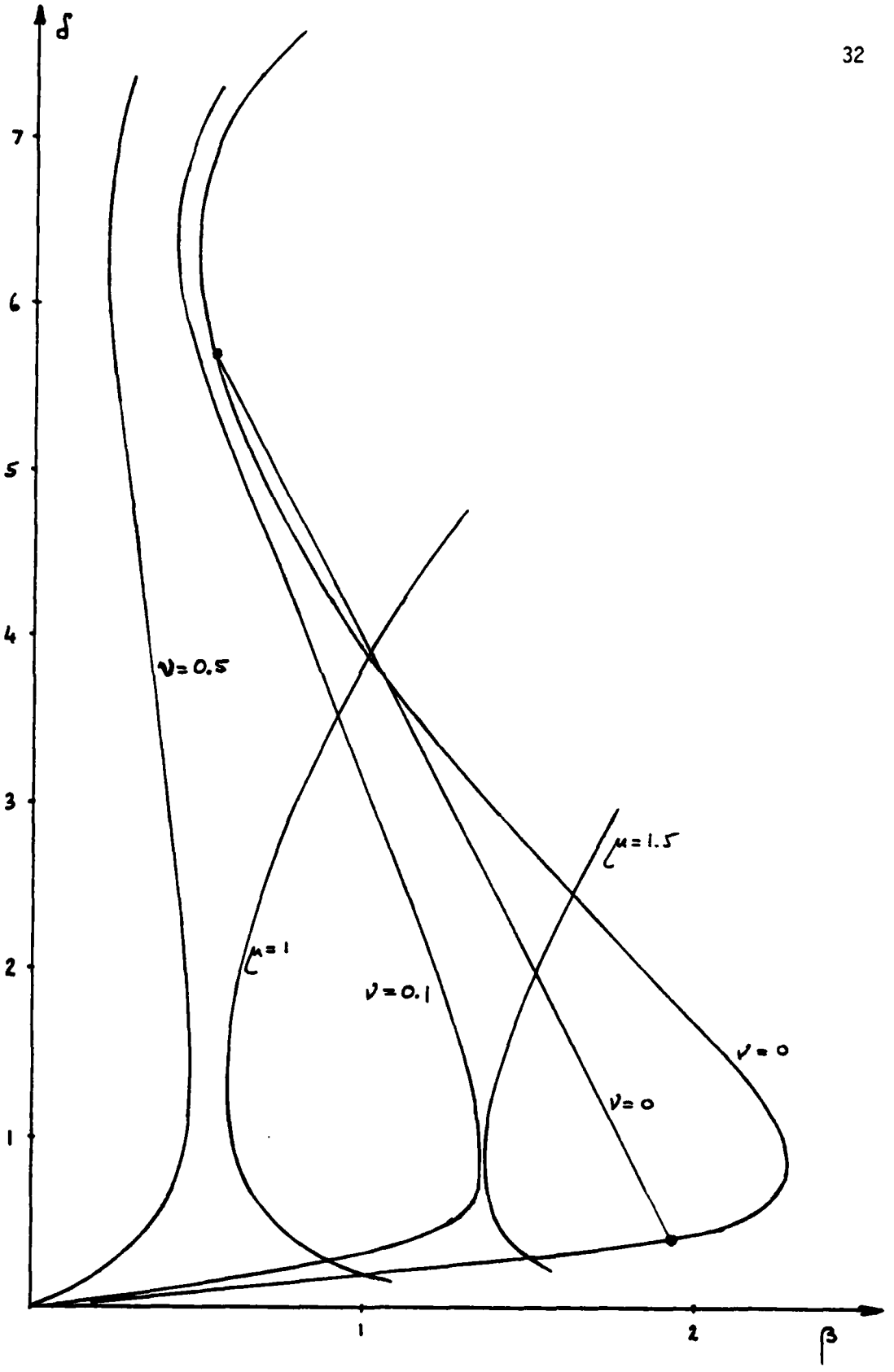


Figure 4

is given by

$$(6.2)\qquad \int_{-\theta_o}^{\theta_o} \{[(\frac{dw}{d\theta} - u) + \frac{1}{2}\frac{H}{R}(\frac{du}{d\theta})^2]^2 + \alpha_1 (\frac{d^2u}{d\theta^2})^2 - \alpha_2 p u\}d\theta.$$

Here $p = p(\theta)$ is the dimensionless radial load, and α_1, α_2 are dimensionless constants. Each end is assumed to be pinned, that is, we have the boundary conditions

$$(6.3)\qquad u(\pm \theta_o) = 0, \quad w(\pm \theta_o) = 0, \quad \frac{d^2u}{d\theta^2}(\pm \theta_o) = 0.$$

The finite element approximation introduced in [36] was applied. More specifically, we used a uniform mesh with eight elements, $\theta_o = 15$ and the constants $\alpha_1 = 3.8716 \times 10^{-6}$, $\alpha_2 = 1.65504 \times 10^{-1}$ corresponding to the data in [16]. Moreover, the following load function $p = p(\mu,\nu)$ was chosen

$$(6.4)\qquad p(\mu,\nu) = \begin{cases} \mu(1 + 7\nu), & \text{for element 4} \\ \mu(1 - \nu), & \text{otherwise} \end{cases}$$

corresponding to a base load $\beta = \mu(1-\nu)$ and an excess load $8\mu\nu$ in element 4 such that the average load is always μ.

Several curves on the equilibrium surface corresponding to constant values of μ or ν were computed. Figure 4 shows the projection of these curves into the (β,δ)-plane where δ represents the radial displacement of the center point. For uniform loads, that is, $\nu = 0$, we encounter two bifurcation points on the primary curve which are connected by two "buckling" curves that have the same projection in the (β,δ)-plane.

7. THE PITCON CODE

```
      SUBROUTINE PITCON(NVAR,LIM,IT,XIT,KSTEP,IPC,H,IRET,IMOD,IPUT,     PTCN0001
     1 HMAX,HMIN,HFACT,ABSERR,RELERR,RWORK,ISIZE)                       PTCN0002
C                                                                       PTCN0003
C***********************************************************************PTCN0004
C                                                                       PTCN0005
C  PITCON.FOR                                                           PTCN0006
C                                                                       PTCN0007
C  29 OCT 1981 VERSION OF PITCON,                                       PTCN0008
C  THE UNIVERSITY OF PITTSBURGH CONTINUATION PACKAGE.                   PTCN0009
C  THIS VERSION USES SINGLE PRECISION AND FULL MATRIX STORAGE.          PTCN0010
C                                                                       PTCN0011
C                                                                       PTCN0012
C  THIS PACKAGE WAS PREPARED WITH THE PARTIAL SUPPORT OF                PTCN0013
C  THE NATIONAL SCIENCE FOUNDATION, UNDER GRANT MCS-78-05299,           PTCN0014
C                                                                       PTCN0015
C  BY WERNER C. RHEINBOLDT AND JOHN V. BURKARDT                         PTCN0016
C  INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS             PTCN0017
C  DEPARTMENT OF MATHEMATICS AND STATISTICS,                            PTCN0018
C  UNIVERSITY OF PITTSBURGH, PITTSBURGH, PA 15261.                      PTCN0019
C                                                                       PTCN0020
C                                                                       PTCN0021
C  AN EARLIER VERSION OF THE PACKAGE WAS WRITTEN IN COOPERATION WITH    PTCN0022
C                                                                       PTCN0023
C  GEORGE D BYRNE,                                                      PTCN0024
C  COMPUTING TECHNOLOGY AND SERVICES                                    PTCN0025
C  EXXON RESEARCH AND ENGINEERING COMPANY                               PTCN0026
C  LINDEN, NEW JERSEY, 07036                                            PTCN0027
C                                                                       PTCN0028
C                                                                       PTCN0029
C  THIS PACKAGE COMPUTES POINTS ALONG A SOLUTION CURVE OF AN            PTCN0030
C  UNDERDETERMINED SYSTEM OF NONLINEAR EQUATIONS OF THE FORM FX=0.      PTCN0031
C  THE CURVE IS SPECIFIED TO PASS THROUGH A GIVEN STARTING SOLUTION     PTCN0032
C  X OF THE SYSTEM.  HERE X DENOTES A REAL VECTOR OF NVAR               PTCN0033
C  COMPONENTS AND FX A REAL VECTOR OF NVAR-1 COMPONENTS.                PTCN0034
C  NORMALLY EACH CALL TO PITCON PRODUCES A NEW POINT FURTHER ALONG      PTCN0035
C  THE SOLUTION CURVE IN A USER-SPECIFIED DIRECTION.                    PTCN0036
C    AN OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF TARGET POINTS,  PTCN0037
C  THAT IS, SOLUTION POINTS X FOR WHICH X(IT) = XIT FOR SOME USER       PTCN0038
C  SPECIFIED VALUES OF IT AND XIT.                                      PTCN0039
C    A FURTHER OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF LIMIT    PTCN0040
C  POINTS FOR SPECIFIED COORDINATE LIM, THAT IS, SOLUTION POINTS FOR    PTCN0041
C  WHICH THE LIM-TH COMPONENT OF THE TANGENT VECTOR IS ZERO.            PTCN0042
C                                                                       PTCN0043
C                                                                       PTCN0044
C  EXPLANATIONS OF THE ALGORITHMS USED IN THIS PACKAGE MAY              PTCN0045
C  BE FOUND IN THE FOLLOWING REFERENCES:                                PTCN0046
C                                                                       PTCN0047
C                                                                       PTCN0048
C  WERNER RHEINBOLDT,                                                   PTCN0049
C  SOLUTION FIELD OF NONLINEAR EQUATIONS AND CONTINUATION METHODS       PTCN0050
C  SIAM JOURNAL OF NUMERICAL ANALYSIS, 17, 1980, PP 221-237             PTCN0051
C                                                                       PTCN0052
C  COR DEN HEIJER AND WERNER RHEINBOLDT,                                PTCN0053
C  ON STEPLENGTH ALGORITHMS FOR A CLASS OF CONTINUATION METHODS,        PTCN0054
C  SIAM JOURNAL OF NUMERICAL ANALYSIS 18, 1981, PP 925-947              PTCN0055
C                                                                       PTCN0056
C  WERNER RHEINBOLDT,                                                   PTCN0057
C  NUMERICAL ANALYSIS OF CONTINUATION METHODS FOR NONLINEAR             PTCN0058
C  STRUCTURAL PROBLEMS,                                                 PTCN0059
C  COMPUTERS AND STRUCTURES, 13, 1981, PP 103-114                       PTCN0060
C                                                                       PTCN0061
C                                                                       PTCN0062
C  OUTLINE OF THE MATHEMATICAL PROCEDURE                                PTCN0063
C                                                                       PTCN0064
C                                                                       PTCN0065
C    THE FUNCTION F DEFINING THE SYSTEM OF EQUATIONS IS SUPPOSED TO     PTCN0066
C  BE CONTINUOUSLY DIFFERENTIABLE.  THE REGULARITY SET R(F) OF F IS     PTCN0067
C  THE SET OF ALL POINTS X IN NVAR-DIMENSIONAL SPACE WHERE THE          PTCN0068
C  DERIVATIVE DF(X) HAS MAXIMAL RANK.  THE STARTING POINT IS ASSUMED    PTCN0069
C  TO BE IN R(F).  FOR ANY INDEX IP WITH 1.LE.IP.LE.NVAR, LET           PTCN0070
```

```
C  FA(X,IP) BE THE FUNCTION OBTAINED BY AUGMENTING F WITH                PTCN0071
C  AN NVAR-TH SCALAR FUNCTION X(IP)-B FOR SOME NUMBER B.  FOR ANY         PTCN0072
C  X IN R(F) THERE IS AT LEAST ONE INDEX IP SUCH THAT THE DERIVATIVE      PTCN0073
C  DFA(X,IP) OF FA IS NONSINGULAR.  WITH SUCH AN IP, AND A DIRECTION      PTCN0074
C  DIR (=+1 OR -1), THE TANGENT OF FA, TAN(X), IS UNIQUELY DEFINED BY:    PTCN0075
C                                                                         PTCN0076
C    TAN:= SOLUTION OF (  DFA(X,IP)*TAN=E(NVAR)  )                       PTCN0077
C    TAN:= TAN/(EUCLIDEAN NORM OF TAN)                                   PTCN0078
C    SN:=  DIR*SIGN (DETERMINANT (DFA(X,IP) ) )                          PTCN0079
C    TAN:= SN*TAN                                                        PTCN0080
C                                                                        PTCN0081
C    HERE E(I) IS THE I-TH BASIS VECTOR IN NVAR-SPACE.                   PTCN0082
C    THE PROCESS USES A LOCAL PARAMETERIZATION OF THE CURVE.             PTCN0083
C  NORMALLY THE CONTINUATION PARAMETER INDEX IP IS CHOSEN AS             PTCN0084
C  THAT INDEX FOR WHICH ABS(TAN(X)(IP)) IS MAXIMAL.  BUT IN THE          PTCN0085
C  CASE OF CERTAIN CURVATURE CHANGES WHERE IT APPEARS THAT               PTCN0086
C  A LIMIT POINT FOR THIS CHOICE FOR IP IS APPROACHING,                  PTCN0087
C  OTHER CHOICES FOR IP MAY BE USED.                                     PTCN0088
C    PREDICTION TAKES PLACE ALONG THE EULER LINE X+H*TAN.  THE           PTCN0089
C  STEPLENGTH ALGORITHM TAKES INTO ACCOUNT THE QUALITY OF THE            PTCN0090
C  CORRECTOR ITERATION AT THE LAST POINT AND A PREDICTION OF THE         PTCN0091
C  CHANGE IN CURVATURE.  THE TANGENTIAL STEPSIZE USED IN PREDICTION IS   PTCN0092
C  CHOSEN SO AS TO ACHIEVE APPROXIMATELY THE PREDICTED SECANT STEPSIZE   PTCN0093
C  AFTER CORRECTION IS DONE.                                             PTCN0094
C    THE CORRECTOR ITERATION STARTS FROM THE PREDICTED POINT AND SOLVES  PTCN0095
C  THE AUGMENTED SYSTEM FA(X,IP)=0 WITH THE VALUE OF THE SCALAR          PTCN0096
C  B EQUAL TO THE IP-TH COMPONENT OF THE PREDICTED POINT.  THE USER      PTCN0097
C  CAN SPECIFY AS CORRECTOR ITERATION EITHER A FULL NEWTON PROCESS,      PTCN0098
C  OR A MODIFIED NEWTON PROCESS WITH FIXED JACOBIAN DFA EVALUATED AT     PTCN0099
C  THE PREDICTED POINT.                                                  PTCN0100
C                                                                        PTCN0101
C                                                                        PTCN0102
C  OUTLINE OF THE COMPUTATIONAL ALGORITHM                                PTCN0103
C                                                                        PTCN0104
C                                                                        PTCN0105
C  DURING THE FOLLOWING DESCRIPTION, WE WILL ASSUME THAT WE              PTCN0106
C  HAVE ENTERED THE CONTINUATION LOOP WITH AN OLD POINT XL,              PTCN0107
C  A CURRENT POINT XC, THE TANGENT TL AT XL, AND CERTAIN SCALAR          PTCN0108
C  QUANTITIES ASSOCIATED WITH THESE VECTORS.  WE WILL CHECK              PTCN0109
C  FIRST FOR ANY TARGET OR LIMIT POINTS BETWEEN XL AND XC,               PTCN0110
C  THEN PROCEED TO COMPUTE A NEW CONTINUATION POINT XF.                  PTCN0111
C  THESE NAMES ARE NOT IN PRECISE ACCORDANCE WITH THE STORAGE            PTCN0112
C  ARRANGEMENTS UNTIL THE END OF A CONTINUATION STEP.                    PTCN0113
C                                                                        PTCN0114
C                                                                        PTCN0115
C  STEP 1:  FOR KSTEP.GT.0, THE CODE GOES TO STEP 2.                     PTCN0116
C           ON THE FIRST CALL TO PITCON FOR A GIVEN PROBLEM (KSTEP=-1    PTCN0117
C           OR KSTEP=0) PROBLEM-DEPENDENT CONSTANTS ARE SET              PTCN0118
C           AND USER CONTROL PARAMETERS ARE LOADED OR DEFAULTS USED.     PTCN0119
C           IF (KSTEP.EQ.0), THE PROGRAM PROCEEDS TO STEP 2.             PTCN0120
C           IF (KSTEP.EQ.-1), THE USER REQUESTS THAT THE INPUT STARTING  PTCN0121
C           POINT XR BE CHECKED FOR THE CONDITION                        PTCN0122
C           ABS(F(XR)).LE.(ABSERR/2).  IF THIS IS NOT THE CASE, NEWTON'SPTCN0123
C           METHOD IS APPLIED TO THE POINT XR UNTIL THE ERROR CONDITION  PTCN0124
C           IS SATISFIED, OR A FAILURE HAS OCCURRED.  AN UNIMPROVABLE    PTCN0125
C           POINT RESULTS IN A RETURN OF IRET=-6.                        PTCN0126
C           IF THE STARTING POINT XR WAS IMPROVED, THE PROGRAM RETURNS   PTCN0127
C           WITH IRET=0 AND KSTEP=0.                                     PTCN0128
C           IF (KSTEP.EQ.0), THE CONTINUATION LOOP BEGINS WITH THE       PTCN0129
C           STARTING POINT XR STORED IN XL AND XC, THE STEPSIZE HTANCF   PTCN0130
C           SET TO THE INPUT VALUE OF H, AND THE                         PTCN0131
C           CONTINUATION PARAMETER SET TO THE INPUT VALUE OF IPC.        PTCN0132
C           FOR KSTEP.GT.0, THESE QUANTITIES ARE COMPUTED AND            PTCN0133
C           UPDATED BY THE PROGRAM ITSELF.                               PTCN0134
C                                                                        PTCN0135
C  STEP 2:  TARGET POINT CHECK.  IF (IT.NE.0), A TARGET POINT IS         PTCN0136
C           DESIRED.  THE VALUES OF XL(IT) AND XC(IT) ARE COMPARED TO    PTCN0137
C           XIT.  IF THE TARGET POINT IS BETWEEN XL AND XC, THE PROGRAM  PTCN0138
C           COMPUTES THE TARGET POINT, SETS IRET=1, AND                  PTCN0139
C           RETURNS, TEMPORARILY INTERRUPTING NORMAL CONTINUATION.       PTCN0140
```

```
C                                                                       PTCN0141
C  STEP 3:  TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION. IF THEPTCN0142
C           LOOP WAS SUSPENDED AT THE LAST CALL TO PITCON TO ALLOW THE  PTCN0143
C           RETURN OF A LIMIT POINT, THEN THE TANGENT HAS ALREADY BEEN  PTCN0144
C           CALCULATED AND A LIMIT POINT CHECK IS SUPERFLUOUS, SO       PTCN0145
C           THE PROGRAM SKIPS TO STEP 5.                                PTCN0146
C           OTHERWISE, A VECTOR IN THE TANGENT PLANE AT XC IS COMPUTED. PTCN0147
C           SUPPOSE THAT THE PREVIOUS CONTINUATION PARAMETER INDEX      PTCN0148
C           WAS IPL, WHERE ON THE FIRST STEP IPL IS USER-SUPPLIED.      PTCN0149
C           THE NEW TANGENT IS NORMALIZED, AND THE IPL-TH COMPONENT     PTCN0150
C           IS FORCED TO HAVE THE SAME SIGN AS THE IPL-TH COMPONENT     PTCN0151
C           OF THE PREVIOUS TANGENT (OR ON FIRST STEP, THE SAME         PTCN0152
C           SIGN AS THE USER INPUT DIRECTION DIR.)  THEN THE LOCAL      PTCN0153
C           CONTINUATION PARAMETER IPC IS DETERMINED.  IPC IS SET TO THEPTCN0154
C           LOCATION OF THE LARGEST COMPONENT OF THE TANGENT VECTOR     PTCN0155
C           UNLESS A LIMIT POINT FOR THIS CHOICE APPEARS TO BE          PTCN0156
C           APPROACHING, IN WHICH CASE THE LOCATION OF THE SECOND       PTCN0157
C           LARGEST COMPONENT MAY BE TRIED.                             PTCN0158
C           ONCE IPC IS SET, THE QUANTITIES TCIPC, TCLIM, HSFCLC, ALPHLCPTCN0159
C           AND DIR ARE COMPUTED, WHOSE MEANINGS ARE EXPLAINED BELOW.   PTCN0160
C                                                                       PTCN0161
C  STEP 4:  LIMIT POINT CHECK.  IF LIM.NE.0, THE LIM-TH COMPONENTS      PTCN0162
C           OF THE OLD AND NEW TANGENTS ARE COMPARED.  IF THESE DIFFER  PTCN0163
C           IN SIGN, A LIMIT POINT LIES BETWEEN XL AND XC.  THE PROGRAM PTCN0164
C           ATTEMPTS TO FIND THIS LIMIT POINT.  IF FOUND, IT STORES     PTCN0165
C           THE LIMIT POINT IN XR, THE TANGENT AT XR IN TL, SETS IRET=2,PTCN0166
C           AND RETURNS, TEMPORARILY INTERRUPTING THE NORMAL LOOP.      PTCN0167
C                                                                       PTCN0168
C  STEP 5:  STEP LENGTH COMPUTATION.  THE PROGRAM COMPUTES HTANCF, THE  PTCN0169
C           STEPSIZE TO BE USED ALONG THE TANGENT TO OBTAIN THE         PTCN0170
C           PREDICTED POINT XPRED:=XC+HTANCF*TC, THE STARTING POINT     PTCN0171
C           FOR THE CORRECTOR PROCESS.  IN COMPUTING HTANCF, CERTAIN    PTCN0172
C           CURVATURE AND STEPSIZE DATA ARE UPDATED.                    PTCN0173
C                                                                       PTCN0174
C  STEP 6:  PREDICTION AND CORRECTION STEP.  WITH THE PREDICTED POINT   PTCN0175
C           XPRED=XC+HTANCF*TC AS A STARTING POINT, THE CORRECTOR       PTCN0176
C           PROCESS IS APPLIED TO CORRECT THE POINT UNTIL               PTCN0177
C           ABS(F(XCOR)).LE.ABSERR AND XSTEP, THE LAST CORRECTOR STEP,  PTCN0178
C           SATISFIES XSTEP.LE.ABSERR+RELERR*ABS(XCOR).                 PTCN0179
C           IF THE SIZE OF A CORRECTOR STEP IS TOO LARGE,               PTCN0180
C           OR IF A CORRECTION STEP INCREASES THE FUNCTION VALUE, OR    PTCN0181
C           THE MAXIMUM NUMBER OF STEPS ARE TAKEN WITHOUT CONVERGENCE,  PTCN0182
C           THE STEPSIZE HTANCF IS REDUCED AND THE CORRECTOR STEP IS    PTCN0183
C           ATTEMPTED AGAIN.  IF THE STEPSIZE SHRINKS BELOW HMIN, THE   PTCN0184
C           PROGRAM SETS AN ERROR FLAG AND RETURNS.                     PTCN0185
C                                                                       PTCN0186
C  STEP 7:  STORING INFORMATION BEFORE RETURN.  AFTER A SUCCESSFUL      PTCN0187
C           CONTINUATION STEP, THE PROGRAM REARRANGES ITS STORAGE SO    PTCN0188
C           THAT THE ENTRIES CORRESPONDING TO XC AND XF HOLD THE PROPER PTCN0189
C           DATA, COMPUTES CORDIS, THE SIZE OF THE CORRECTION TO THE    PTCN0190
C           PREDICTED POINT, AND MODIFIES CORDIS TO A VALUE THAT WOULD  PTCN0191
C           CORRESPOND TO AN OPTIMAL NUMBER OF CORRECTOR STEPS.         PTCN0192
C                                                                       PTCN0193
C                                                                       PTCN0194
C  ON NORMAL RETURN, THE VECTOR XR (THE FIRST NVAR ENTRIES OF RWORK),   PTCN0195
C  CONTAINS A SOLUTION POINT ON THE CURVE F(XR)=0, AND IS EITHER        PTCN0196
C  A CONTINUATION POINT, A TARGET POINT, OR A LIMIT POINT, WHICH        PTCN0197
C  IS INDICATED BY THE VALUE OF IRET.                                   PTCN0198
C  IF IRET IS NEGATIVE, AN ERROR HAS OCCURRED.  IF A LIMIT POINT IS     PTCN0199
C  RETURNED, THE TANGENT VECTOR AT THE LIMIT POINT IS CONTAINED IN THE  PTCN0200
C  LOCATION TL IN RWORK.  ON FIRST CALL, THE USER MUST SET SOME OF THE  PTCN0201
C  SCALAR PARAMETERS, AND THE STARTING POINT XR.  THEREAFTER, ONLY IT   PTCN0202
C  AND XIT SHOULD BE CHANGED BY THE USER DURING A PROBLEM RUN.          PTCN0203
C    IF A NEW PROBLEM IS TO BE RUN (WHETHER A DIFFERENT FUNCTION, OR THEPTCN0204
C  SAME FUNCTION WITH DIFFERENT STARTING POINT OR ERROR CONTROLS)       PTCN0205
C  THE PROGRAM MAY BE RESET BY USING KSTEP=-1 OR 0, AT WHICH TIME THE   PTCN0206
C  SCALARS AND THE POINT XR MUST BE SET AGAIN.  NOTE THAT IN THIS CASE  PTCN0207
C  THE STATISTICAL DATA IN THE COMMON BLOCKS /COUNT1/ AND /COUNT2/      PTCN0208
C  WILL BE RESET TO 0 AS WELL.                                          PTCN0209
C                                                                       PTCN0210
```

```
C                                                                       PTCN0211
C  DEFINITIONS AND DEFAULTS OF PITCON PARAMETERS                        PTCN0212
C                                                                       PTCN0213
C                                                                       PTCN0214
C  NVAR   = THE NUMBER OF VARIABLES IN THE NONLINEAR SYSTEM.  NVAR IS   PTCN0215
C           THE DIMENSION OF THE PIVOT VECTOR IPVT, AND THE SIZE OF THE PTCN0216
C           VECTORS XR, XC, XF, TL AND TC WHICH ARE CONTAINED IN RWORK. PTCN0217
C           RWORK ALSO CONTAINS STORAGE FOR THE MATRIX FPRYM WHICH      PTCN0218
C           IS OF SIZE NVAR X NVAR.                                     PTCN0219
C           NVAR MUST BE GREATER THAN 1, AND MUST NOT BE CHANGED DURING PTCN0220
C           THE COURSE OF A PROBLEM RUN.  NVAR HAS NO DEFAULT VALUE.    PTCN0221
C  LIM    = LIMIT POINT FLAG AND INDEX.  IF (LIM.EQ.0), LIMIT POINTS   PTCN0222
C           ARE NOT TO BE LOOKED FOR.  OTHERWISE, THE USER SHOULD SET   PTCN0223
C           LIM TO THE INDEX OF THE VARIABLE FOR WHICH LIMIT            PTCN0224
C           POINTS ARE TO BE SOUGHT.  LIM DEFAULTS TO ZERO.             PTCN0225
C           LIM MUST SATISFY 0.LE.LIM.LE.NVAR.                          PTCN0226
C  IT     = TARGET POINT FLAG AND INDEX.  IF (IT.EQ.0), TARGET POINTS   PTCN0227
C           ARE NOT TO BE LOOKED FOR.  OTHERWISE, THE USER SHOULD SET   PTCN0228
C           IT TO THE INDEX OF THE VARIABLE FOR WHICH TARGET            PTCN0229
C           VALUES XIT ARE DESIRED.  IT HAS THE DEFAULT VALUE ZERO.     PTCN0230
C           IT MAY BE RESET BY THE USER AT ANY TIME DURING A RUN.       PTCN0231
C           IT MUST SATISFY 0.LE.IT.LE.NVAR.                            PTCN0232
C  XIT    = THE VALUE OF THE TARGET VECTOR COMPONENT SOUGHT, IF IT.NE.0. PTCN0233
C           TARGET POINTS XR SATISFY XR(IT)=XIT.  XIT HAS NO DEFAULT.   PTCN0234
C  KSTEP  = THE NUMBER OF CONTINUATION STEPS TAKEN.  THIS DOES NOT      PTCN0235
C           INCLUDE FAILURES, TARGET POINTS OR LIMIT POINTS.  THE PROGRAMPTCN0236
C           INCREMENTS KSTEP EACH TIME A NEW POINT XF IS COMPUTED.      PTCN0237
C           ON THE FIRST CALL TO PITCON FOR A PARTICULAR PROBLEM, THE   PTCN0238
C           USER SHOULD SET KSTEP TO 0 OR -1.  IF KSTEP=-1, THE PROGRAM PTCN0239
C           WILL CHECK THE ACCURACY OF THE STARTING POINT XR, AND IF    PTCN0240
C           NECESSARY, ATTEMPT TO CORRECT IT USING NEWTON'S METHOD.     PTCN0241
C           IF KSTEP=0, THE PROGRAM PERFORMS NO CHECK ON THE STARTING   PTCN0242
C           POINT, AND PROCEEDS TO THE CONTINUATION LOOP.  IF THE USER  PTCN0243
C           WISHES TO RUN A DIFFERENT PROBLEM THEN A CALL TO PITCON     PTCN0244
C           WITH KSTEP=-1 OR 0 WILL RESET THE CODE, DESTROYING THE      PTCN0245
C           INFORMATION FROM THE PREVIOUS RUN.  KSTEP DEFAULTS TO -1.   PTCN0246
C  IPC    = THE COMPONENT OF XC TO BE USED AS CONTINUATION PARAMETER.   PTCN0247
C           ON THE FIRST CALL ONLY, THE USER OUGHT TO SET IPC OR ALLOW  PTCN0248
C           THE DEFAULT VALUE IPC=NVAR.  AFTER THE FIRST CALL, THE      PTCN0249
C           DETERMINATION OF IPC IS DONE BY THE PROGRAM USING INFORMATIONPTCN0250
C           ABOUT THE TANGENT VECTOR AT XC.                             PTCN0251
C  H      = SUGGESTED STARTING STEP SIZE ALONG THE TANGENT TO THE CURVE. PTCN0252
C           IF H=0.0 ON THE INITIAL CALL, H DEFAULTS TO (HMAX+HMIN)/2   PTCN0253
C           IF H IS NEGATIVE ON THE FIRST CALL, THE MINUS SIGN          PTCN0254
C           IS ABSORBED BY DIR AND INDICATES THAT THE DIRECTION OF THE  PTCN0255
C           FIRST STEP SHOULD BE IN THE NEGATIVE IPC DIRECTION.         PTCN0256
C           AFTER THE FIRST STEP, STEPSIZE IS CONTROLLED BY THE PROGRAM. PTCN0257
C           UPON RETURN WITH A CONTINUATION POINT (IRET=0), H IS        PTCN0258
C           OVERWRITTEN BY HTANCF, THE STEPSIZE USED IN REACHING THE    PTCN0259
C           NEW POINT.                                                  PTCN0260
C  IRET   = A RETURN FLAG TO INDICATE ERRORS OR THE TYPE OF POINT       PTCN0261
C           RETURNED IN XR.  NONNEGATIVE VALUES OF IRET REPRESENT       PTCN0262
C           NORMAL RETURNS.  NEGATIVE VALUES OF IRET INDICATE THAT      PTCN0263
C           SOME ERROR OR DIFFICULTY HAS BEEN ENCOUNTERED.  VALUES      PTCN0264
C           OF IRET BETWEEN -1 AND -4 ARE SIMPLY REPORTS THAT AN        PTCN0265
C           ATTEMPT TO COMPUTE A LIMIT OR TARGET POINT FAILED.  THESE   PTCN0266
C           DO NOT AFFECT FURTHER CONTINUATION STEPS, AND THE USER NEED PTCN0267
C           NOT MODIFY ANY VARIABLES BEFORE PROCEEDING.                 PTCN0268
C           VALUES OF IRET OF -5 AND -6 REFER TO DANGEROUS SITUATIONS   PTCN0269
C           THAT MAY BE CORRECTABLE.                                    PTCN0270
C           VALUES OF IRET FROM -7 TO -10 ARE SERIOUS, FATAL ERRORS.    PTCN0271
C           THE USER SHOULD HALT THE PROGRAM AND EXAMINE HIS INPUT      PTCN0272
C           AND THE INTERIM RESULTS.                                    PTCN0273
C           IRET SHOULD BE ZERO ON THE FIRST CALL FOR A PROBLEM.        PTCN0274
C           THE SPECIFIC VALUES OF IRET AND THEIR MEANINGS ARE:         PTCN0275
C                                                                       PTCN0276
C           IRET=2:   NORMAL RETURN WITH LIMIT POINT IN XR AND TANGENT  PTCN0277
C                     AT XR CONTAINED IN TL.                            PTCN0278
C                                                                       PTCN0279
C           IRET=1:   NORMAL RETURN WITH TARGET POINT IN XR.            PTCN0280
```

```
C                                                                       PTCN0281
C          IRET=0:   NORMAL RETURN WITH NEW CONTINUATION POINT IN XR.   PTCN0282
C                                                                       PTCN0283
C          IRET=-1:  AN ERROR OCCURRED DURING COMPUTATION OF LIMIT POINTPTCN0284
C                                                                       PTCN0285
C          IRET=-2:  CORECT CALLED FOR TARGET POINT CALCULATION FAILED  PTCN0286
C                    AFTER KNMAX ITERATIONS.                            PTCN0287
C                                                                       PTCN0288
C          IRET=-3:  SOLVE WAS CALLED BY CORECT FOR TARGET POINT        PTCN0289
C                    CALCULATION, AND FAILED.  (MATRIX ELIMINATION FOUNDPTCN0290
C                    ZERO PIVOT).                                       PTCN0291
C                                                                       PTCN0292
C          IRET=-4:  UNACCEPTABLE CORRECTOR STEP IN TARGET POINT        PTCN0293
C                    CALCULATION.                                       PTCN0294
C                                                                       PTCN0295
C          IRET=-5:  PREDICTION STEP HTANCF IS LESS THAN HMIN, PERHAPS  PTCN0296
C                    BECAUSE OF REPEATED FAILURE OF CORECT, AND         PTCN0297
C                    CONSEQUENT STEPSIZE REDUCTION.  USER MIGHT REDUCE  PTCN0298
C                    HMIN, OR SWITCH FROM IMOD=1 TO IMOD=0, OR INCREASE PTCN0299
C                    ABSERR AND RELERR.  BUT BE AWARE THAT REPEATED     PTCN0300
C                    STEPSIZE REDUCTIONS MAY INDICATE AN INTRACTABLE    PTCN0301
C                    FUNCTION.                                          PTCN0302
C                                                                       PTCN0303
C          IRET=-6:  FUNCTION VALUE FNRMXF OF INPUT XR IS TOO LARGE     PTCN0304
C                    AND COULD NOT BE IMPROVED BY CORECT.  USER         PTCN0305
C                    MIGHT RECOVER BY RELAXING ERROR CONTROLS, IMPROVINGPTCN0306
C                    STARTING POINT XR, OR CHANGING VALUE OF IPC.       PTCN0307
C                                                                       PTCN0308
C          IRET=-7:  SOLVE FAILED IN A CALL FROM TANGNT.                PTCN0309
C                                                                       PTCN0310
C          IRET=-8:  SOLVE FAILED IN A CALL FROM CORECT.                PTCN0311
C                                                                       PTCN0312
C          IRET=-9:  THE TANGENT VECTOR TC AT XC IS ZERO.               PTCN0313
C                                                                       PTCN0314
C          IRET=-10: IMPROPER INPUT, NVAR.LE.1, OR                      PTCN0315
C                    ISIZE.LT.(NVAR)*(NVAR+5), OR                       PTCN0316
C                    PROGRAM HAS BEEN CALLED AGAIN AFTER FATAL ERROR.   PTCN0317
C                                                                       PTCN0318
C  IMOD   = METHOD FLAG FOR CORRECTOR STEP, SPECIFYING TYPE OF          PTCN0319
C           NEWTON METHOD TO BE USED.                                   PTCN0320
C                                                                       PTCN0321
C           IMOD=0: UPDATE JACOBIAN FOR TANGENT CALCULATION,            PTCN0322
C                   UPDATE JACOBIAN FOR EACH CORRECTOR STEP.            PTCN0323
C                                                                       PTCN0324
C           IMOD=1: UPDATE JACOBIAN FOR TANGENT CALCULATION,            PTCN0325
C                   EVALUATE JACOBIAN AT FIRST CORRECTOR STEP ONLY.     PTCN0326
C                                                                       PTCN0327
C  IPVT   = INTEGER VECTOR USER DECLARED TO BE OF SIZE NVAR.            PTCN0328
C           USED DURING THE MATRIX FACTORIZATION TO STORE PIVOT         PTCN0329
C           INFORMATION.                                                PTCN0330
C  HMAX   = THE MAXIMUM STEP SIZE. IF HMAX.LE.0.0 ON INITIAL CALL,      PTCN0331
C           HMAX DEFAULTS TO SQRT(NVAR).                                PTCN0332
C  HMIN   = THE MINIMUM STEP SIZE. IF HMIN.LE.SQRT(EPMACH) ON INITIAL   PTCN0333
C           CALL, HMIN DEFAULTS TO SQRT(EPMACH), WHERE EPMACH IS THE    PTCN0334
C           MACHINE PRECISION CONSTANT.                                 PTCN0335
C  HFACT  = LIMIT ON STEPSIZE CHANGE.  HSECLC IS THE SECANT STEPSIZE    PTCN0336
C           OF THE LAST STEP, AND HTANCF IS THE STEPSIZE TO BE USED     PTCN0337
C           IN OBTAINING THE PREDICTED POINT.  THE FOLLOWING RELATIONSHIPPTCN0338
C           MUST BE SATISFIED: (HSECLC/HFACT).LE.HTANCF.LE.(HSECLC*HFACT)PTCN0339
C           IF THE CORRECTOR STEP FAILS, THEN HFACT IS ALSO USED TO     PTCN0340
C           REDUCE THE PREDICTOR STEP HTANCF TO (HTANCF/HFACT)          PTCN0341
C           IF HFACT.LE.1.0 ON INITIAL CALL, HFACT DEFAULTS TO 3.0.     PTCN0342
C  ABSERR= ABSOLUTE ERROR CONTROL. IF ABSERR=0.0 ON INITIAL CALL        PTCN0343
C           ABSERR DEFAULTS TO SQRT(EPMACH)                             PTCN0344
C  RELERR= RELATIVE ERROR CONTROL. IF RELERR=0.0 ON INITIAL CALL        PTCN0345
C           RELERR DEFAULTS TO SQRT(EPMACH)                             PTCN0346
C  RWORK  = USER DECLARED VECTOR OF SIZE ISIZE=NVAR*(NVAR+5).           PTCN0347
C           RWORK STORES FIVE VECTORS AND AN ARRAY IN THE ORDER         PTCN0348
C           (XR,XC,XF,TL,TC,FPRYM).  THEIR BEGINNING LOCATIONS          PTCN0349
C           ARE IXR=1, IXC=NVAR+1, IXF=2*NVAR+1, ITL=3*NVAR+1,          PTCN0350
C           ITC=4*NVAR+1, IFP=5*NVAR+1.  FPRYM IS AN ARRAY OF           PTCN0351
```

```
C           SIZE NVAR X NVAR.  THE MEANINGS OF THESE COMPONENTS OF       PTCN0352
C           RWORK ARE DESCRIBED BELOW.  THE USER SHOULD SET A VALUE TO XRPTCN0353
C           ON FIRST CALL, BUT NO OTHER PORTIONS OF RWORK SHOULD BE      PTCN0354
C           SET.  AFTER THE FIRST CALL FOR A PROBLEM, NO ENTRIES OF RWORKPTCN0355
C           SHOULD BE ALTERED.                                           PTCN0356
C (XR)    = ON FIRST CALL, A USER SUPPLIED STARTING POINT, WHICH MAY BE  PTCN0357
C           IMPROVED BY THE PROGRAM IF KSTEP=-1.  ON NORMAL RETURN FROM  PTCN0358
C           PITCON, XR WILL HOLD THE MOST RECENTLY FOUND POINT, WHETHER  PTCN0359
C           A CONTINUATION POINT, TARGET POINT, OR LIMIT POINT.          PTCN0360
C (XC)    = THE PREVIOUS CONTINUATION POINT.                             PTCN0361
C (XF)    = THE CURRENT CONTINUATION POINT.                              PTCN0362
C (TL)    = PREVIOUS VALUE OF TANGENT VECTOR.  NOTE THAT THIS CORRESPONDSPTCN0363
C           TO A POINT XL WHICH HAS BEEN DISCARDED.  ON A LIMIT POINT    PTCN0364
C           RETURN, TL WILL CONTAIN INSTEAD THE TANGENT AT THE           PTCN0365
C           LIMIT POINT.  ON A TARGET POINT RETURN, TL WILL HAVE         PTCN0366
C           BEEN OVERWRITTEN BY THE FUNCTION VALUE AT THE TARGET POINT.  PTCN0367
C (TC)    = THE TANGENT VECTOR AT THE PREVIOUS CONTINUATION POINT.       PTCN0368
C (FPRYM)= MATRIX STORAGE AREA FOR SETTING UP AND SOLVING THE            PTCN0369
C           LINEAR SYSTEMS INVOLVING DFA(X,IP).                          PTCN0370
C  ISIZE  = USER SET DIMENSION FOR VECTOR RWORK, WHICH MUST BE AT        PTCN0371
C           LEAST OF SIZE NVAR*(NVAR+5).                                 PTCN0372
C                                                                        PTCN0373
C                                                                        PTCN0374
C  NOMENCLATURE FOR STEP DEPENDENT VARIABLES                             PTCN0375
C                                                                        PTCN0376
C                                                                        PTCN0377
C  THE PROGRAM ACCUMULATES INFORMATION THAT IS ASSOCIATED WITH           PTCN0378
C  SEVERAL PREVIOUS CONTINUATION POINTS OR THE STEPS MADE BETWEEN        PTCN0379
C  THEM.  IN INTERPRETING THE CODE OR ITS OUTPUT, IT IS IMPORTANT        PTCN0380
C  TO KNOW WHERE SUCH QUANTITIES APPLY.  THE FOLLOWING DESCRIPTION       PTCN0381
C  OF SOME OF THE VARIABLES IS VALID ONLY UPON A NORMAL RETURN WITH      PTCN0382
C  A CONTINUATION POINT.                                                 PTCN0383
C  THE POINTS 'XLL' AND 'XL' WILL HAVE BEEN DISCARDED BY THE PROGRAM,    PTCN0384
C  BUT SOME QUANTITIES ASSOCIATED WITH THEM STILL SURVIVE.               PTCN0385
C                                                                        PTCN0386
C  QUANTITIES ASSOCIATED WITH STEP FROM 'XLL' TO 'XL':                   PTCN0387
C                                                                        PTCN0388
C  HSECLL = SIZE OF SECANT FROM 'XLL' TO 'XL', EUCLIDEAN NORM(XLL-XL)    PTCN0389
C                                                                        PTCN0390
C  QUANTITIES ASSOCIATED WITH THE POINT 'XL':                            PTCN0391
C                                                                        PTCN0392
C  IPL    = THE LOCATION OF THE FIRST OR SECOND LARGEST COMPONENT        PTCN0393
C           OF THE TANGENT VECTOR AT 'XL'.                               PTCN0394
C  TLLIM  = VALUE OF LIM-TH COMPONENT OF TANGENT VECTOR AT 'XL'.         PTCN0395
C  TL     = TANGENT VECTOR AT 'XL', ALTHOUGH LIMIT OR TARGET POINT       PTCN0396
C           CALCULATIONS COULD HAVE OVERWRITTEN THIS VECTOR.             PTCN0397
C                                                                        PTCN0398
C  QUANTITIES ASSOCIATED WITH INTERVAL FROM 'XL' TO XC:                  PTCN0399
C                                                                        PTCN0400
C  ALPHLC = THE ANGLE BETWEEN THE TANGENTS AT 'XL' AND XC.               PTCN0401
C  CURVLC = ESTIMATED CURVATURE BETWEEN 'XL' AND XC.                     PTCN0402
C  HSECLC = SIZE OF SECANT BETWEEN 'XL' AND XC, EUCLIDEAN NORM(XL-XC)    PTCN0403
C                                                                        PTCN0404
C  QUANTITIES ASSOCIATED WITH THE POINT XC:                              PTCN0405
C                                                                        PTCN0406
C  DETA   = BINARY MANTISSA OF DETERMINANT OF DFA(XC,IPL), DIVIDED       PTCN0407
C           BY IPL-TH COMPONENT OF TANGENT AT XC.                        PTCN0408
C  DIR    = SIGN OF DETA, DETERMINES SENSE OF CONTINUATION.              PTCN0409
C  IEXP   = BINARY EXPONENT OF DETERMINANT OF DFA(XC,IPL).               PTCN0410
C  IPC    = LOCATION OF FIRST OR SECOND LARGEST COMPONENT OF TANGENT     PTCN0411
C           VECTOR AT XC.                                                PTCN0412
C  TC     = TANGENT VECTOR AT XC.                                        PTCN0413
C  TCLIM  = VALUE OF LIM-TH COMPONENT OF TANGENT AT XC.                  PTCN0414
C  TCIPC  = VALUE OF TC(IPC)                                             PTCN0415
C                                                                        PTCN0416
C  QUANTITIES ASSOCIATED WITH THE INTERVAL FROM XC TO XF:                PTCN0417
C                                                                        PTCN0418
C  CURVCF = ESTIMATED CURVATURE BETWEEN XC AND XF.                       PTCN0419
C  HTANCF = STEPSIZE USED ALONG TANGENT TO GET PREDICTED POINT           PTCN0420
```

```
C           WHICH WAS CORRECTED TO SOLUTION POINT XF.                          PTCN0421
C                                                                              PTCN0422
C  QUANTITIES ASSOCIATED WITH THE POINT XF:                                    PTCN0423
C                                                                              PTCN0424
C  CORDIS = SIZE OF THE TOTAL CORRECTION FROM PREDICTED POINT                  PTCN0425
C           X=XC+HTANGF*TC TO CORRECTED POINT XF.                              PTCN0426
C           NOTE THAT THIS HAS BEEN MODIFIED TO AN 'OPTIMAL' VALUE.            PTCN0427
C  CURVXF = A PREDICTED VALUE OF THE CURVATURE AT XF.                          PTCN0428
C  FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT XF.                              PTCN0429
C  FPRYM  = DFA(XF,IPC) HAS ACTUALLY BEEN LAST EVALUATED AT THE                PTCN0430
C           PENULTIMATE CORRECTOR ITERATE (IF IMOD.NE.1).  IT WILL BE          PTCN0431
C           EVALUATED AT XF AS SOON AS THE NEXT LOOP BEGINS AND THE            PTCN0432
C           TANGENT IS NEEDED.                                                 PTCN0433
C  XSTEP  = SIZE OF THE LAST CORRECTOR STEP TAKEN IN CONVERGING TO XF.         PTCN0434
C                                                                              PTCN0435
C                                                                              PTCN0436
C  SUBROUTINES IN THIS PACKAGE                                                 PTCN0437
C                                                                              PTCN0438
C                                                                              PTCN0439
C  PITCON(NVAR,LIM,IT,XIT,KSTEP,IPC,H,IRET,IMOD,IPVT,                          PTCN0440
C  HMAX,HMIN,HFACT,ABSERR,RELERR,RWORK,ISIZE)                                  PTCN0441
C                                                                              PTCN0442
C  DRIVING ROUTINE OF CONTINUATION CODE.  INITIALIZES INFORMATION,             PTCN0443
C  DETERMINES WHETHER LIMIT, TARGET OR CONTINUATION POINT WILL                 PTCN0444
C  BE SOUGHT THIS STEP, COMPUTES STEPLENGTHS, CONTROLS CORRECTOR               PTCN0445
C  PROCESS, AND HANDLES ERROR RETURNS.                                         PTCN0446
C                                                                              PTCN0447
C  CORECT(NVAR,X,IHOLD,WORK,IERR,IMOD,FPRYM,IPVT,ABSERR,RELERR,                PTCN0448
C  XSTEP,NEQN,FNRM)                                                            PTCN0449
C                                                                              PTCN0450
C  USES A FORM OF NEWTON'S METHOD TO SOLVE THE AUGMENTED NONLINEAR             PTCN0451
C  SYSTEM FA(X)=0 WITH AUGMENTING EQUATION X(IHOLD)=B, THAT IS, X(IHOLD)PTCN0452
C  IS HELD FIXED DURING THE CORRECTION PROCESS.                                PTCN0453
C                                                                              PTCN0454
C  TANGNT(NVAR,XC,IPC,TC,IRET,ICALL,FPRYM,IPVT,NEQN,DETA,IEXP)                 PTCN0455
C                                                                              PTCN0456
C  APPLIES ALGORITHM DESCRIBED ABOVE TO SOLVE DFA(XC,IPL)*TC=E(NVAR)           PTCN0457
C  AND THEN NORMALIZES TANGENT VECTOR, CORRECTS SIGN, AND SETS                 PTCN0458
C  IPC AND DIR.                                                                PTCN0459
C                                                                              PTCN0460
C  ROOT(A,FA,B,FB,C,FC,KOUNT,IFLAG)                                            PTCN0461
C                                                                              PTCN0462
C  ROOT FINDER USED TO LOCATE LIMIT POINT.  THIS ROUTINE IS A MODIFIED         PTCN0463
C  VERSION OF THE FORTRAN FUNCTION ZERO GIVEN IN THE BOOK:                     PTCN0464
C  'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES'                           PTCN0465
C  BY RICHARD P BRENT, PRENTICE HALL, 1973.                                    PTCN0466
C                                                                              PTCN0467
C  SOLVE(NVAR,X,Y,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,IPVT)                     PTCN0468
C                                                                              PTCN0469
C  SETS UP AND SOLVES THE SYSTEM DFA(X,IP)*Y(OUTPUT)=Y(INPUT)                  PTCN0470
C  WHERE DFA(X,IP) IS THE JACOBIAN OF FA AT X,                                 PTCN0471
C  AND Y IS A RIGHT HAND SIDE SUPPLIED BY THE CALLING ROUTINE.                 PTCN0472
C                                                                              PTCN0473
C  **NOTE** SUBROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE             PTCN0474
C  SYSTEM.  THE USER MAY WISH TO REPLACE THIS ROUTINE WITH ONE MORE            PTCN0475
C  SUITED TO HIS PROBLEM.                                                      PTCN0476
C                                                                              PTCN0477
C                                                                              PTCN0478
C  USER SUPPLIED SUBROUTINES                                                   PTCN0479
C                                                                              PTCN0480
C                                                                              PTCN0481
C  FCTN(NVAR,X,FX)                                                             PTCN0482
C                                                                              PTCN0483
C  EVALUATES THE NVAR-1 COMPONENT FUNCTION FX GIVEN X AN NVAR COMPONENT PTCN0484
C  VECTOR. THIS FUNCTION DESCRIBES THE NONLINEAR SYSTEM.  THE AUGMENTINGPTCN0485
C  EQUATION IS HANDLED BY THE CONTINUATION PACKAGE.                            PTCN0486
C                                                                              PTCN0487
C  FPRIME(NVAR,X,FPRYM)                                                        PTCN0488
C                                                                              PTCN0489
C  EVALUATES THE NVAR-1 BY NVAR JACOBIAN MATRIX FPRYM (X)                      PTCN0490
```

```
C     AT X AND STORES IT IN THE NVAR BY NVAR ARRAY FPRYM,                    PTCN0491
C     SO THAT  FPRYM(I,J) CONTAINS (DF(X) (I)/ DX (J)).                      PTCN0492
C     THE LAST ROW OF FPRYM (FOR THE AUGMENTING EQUATION) IS INSERTED        PTCN0493
C     BY THE ROUTINE SOLVE.                                                  PTCN0494
C                                                                            PTCN0495
C                                                                            PTCN0496
C     LINPAK ROUTINES USED                                                   PTCN0497
C                                                                            PTCN0498
C                                                                            PTCN0499
C     LINPAK REFERENCE:                                                      PTCN0500
C     LINPACK USER'S GUIDE,                                                  PTCN0501
C     J J DONGARRA, J R BUNCH, C B MOLER AND G W STEWART,                    PTCN0502
C     SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS,                        PTCN0503
C     PHILADELPHIA, 1979.                                                    PTCN0504
C                                                                            PTCN0505
C                                                                            PTCN0506
C     ISAMAX(N,SX,INCX)                                                      PTCN0507
C     INTEGER FUNCTION RETURNS THE POSITION OF LARGEST ELEMENT OF SX         PTCN0508
C                                                                            PTCN0509
C     SAXPY(N,SA,SX,INCX,SY,INCY)                                            PTCN0510
C     SETS VECTOR SY(I) = SA*SX(I)+SY(I)                                     PTCN0511
C                                                                            PTCN0512
C     SCOPY(N,SX,INCX,SY,INCY)                                               PTCN0513
C     SETS SY(I)=SX(I)                                                       PTCN0514
C                                                                            PTCN0515
C     SDOT(N,SX,INCX,SY,INCY)                                                PTCN0516
C     SDOT = SUM(I=1 TO N) SX(I)*SY(I)                                       PTCN0517
C                                                                            PTCN0518
C     SNRM2(N,SX,INCX)                                                       PTCN0519
C     SNRM2 = EUCLIDEAN NORM OF SX(I)                                        PTCN0520
C                                                                            PTCN0521
C     **NOTE**  SNRM2 HAS MACHINE DEPENDENT CUTOFF CONSTANTS                 PTCN0522
C                                                                            PTCN0523
C     SSCAL(N,SA,SX,INCX)                                                    PTCN0524
C     SETS SX(I)=SA*SX(I)                                                    PTCN0525
C                                                                            PTCN0526
C     SGEFA(A,LDA,N,IPVT,INFO)                                               PTCN0527
C     FACTORS MATRIX A WHOSE LEADING DIMENSION WAS DECLARED AS LDA           PTCN0528
C     AND WHOSE ACTUAL USED DIMENSION IS N, SETS UP PIVOT INFORMATION        PTCN0529
C     IN VECTOR IPVT AND WARNS OF ZERO PIVOTS.                               PTCN0530
C                                                                            PTCN0531
C     SGESL(A,LDA,N,IPVT,B,JOB)                                              PTCN0532
C     ACCEPTS OUTPUT OF SGEFA, AND A RIGHT HAND SIDE B, AND SOLVES           PTCN0533
C     SYSTEM A*X=B, RETURNING X IN B.  FOR MODIFIED NEWTON'S METHOD, ONCE    PTCN0534
C     MATRIX IS FACTORED BY SGEFA, ONLY SGESL IS CALLED FOR SUCCESSIVE       PTCN0535
C     RIGHT HAND SIDES                                                       PTCN0536
C                                                                            PTCN0537
C                                                                            PTCN0538
C                                                                            PTCN0539
C     LABELED COMMON BLOCKS                                                  PTCN0540
C                                                                            PTCN0541
C                                                                            PTCN0542
C     /COUNT1/  COUNTS NUMBER OF CALLS FROM ... TO ... AS FOLLOWS:          PTCN0543
C     ICRSL  = CORECT TO SOLVE                                               PTCN0544
C     ITNSL  = TANGNT TO SOLVE                                               PTCN0545
C     NSTCR  = PITCON TO CORECT FOR IMPROVED STARTING POINT                  PTCN0546
C     NCNCR  = PITCON TO CORECT FOR CONTINUATION POINT                       PTCN0547
C     NTRCR  = PITCON TO CORECT FOR TARGET POINTS                            PTCN0548
C     NLMCR  = PITCON TO CORECT FOR LIMIT POINT                              PTCN0549
C     NLMRT  = PITCON TO ROOT FOR LIMIT POINT                                PTCN0550
C                                                                            PTCN0551
C     NOTE THAT NSTCR, NCNCR, NTRCR, NLMCR AND NLMRT COUNT THE NUMBER        PTCN0552
C     OF ITERATIVE STEPS (NEWTON OR ROOTFINDING) AND NOT JUST THE NUMBER     PTCN0553
C     OF SUBROUTINE CALLS.                                                   PTCN0554
C                                                                            PTCN0555
C     /COUNT2/  KEEPS PERFORMANCE AND WORK STATISTICS                        PTCN0556
C     IFEVAL = NUMBER OF CALLS TO FCTN                                       PTCN0557
C     IPEVAL = NUMBER OF CALLS TO FPRIME                                     PTCN0558
C     ISOLVE = NUMBER OF CALLS TO SOLVE                                      PTCN0559
C     NRED   = NUMBER OF STEPSIZE REDUCTIONS MADE BEFORE PREDICTOR           PTCN0560
```

```
C            POINT CONVERGED TO THE NEW CONTINUATION POINT.                PTCN0561
C  NRDSUM = TOTAL NUMBER OF STEPSIZE REDUCTIONS                             PTCN0562
C  KN      = NUMBER OF CORRECTOR ITERATION STEPS TAKEN IN MOST RECENT        PTCN0563
C            CALL TO CORECT.                                               PTCN0564
C  KNSUM   = TOTAL NUMBER OF CORRECTOR ITERATION STEPS.                     PTCN0565
C                                                                          PTCN0566
C  /OUTPUT/                                                                PTCN0567
C  IWRITE = USER ACCESSIBLE OUTPUT INDICATOR.                              PTCN0568
C           IWRITE=0, NO OUTPUT PRINTED BY PITCON.                         PTCN0569
C           IWRITE=1, ERROR MESSAGES PRINTED BY PITCON.                    PTCN0570
C           IWRITE=2, CERTAIN OUTPUT WILL BE PRINTED BY PITCON.            PTCN0571
C                                                                          PTCN0572
C  /POINT/ CONTAINS DATA ABOUT THE SOLUTION CURVE                          PTCN0573
C  DETA    = BINARY MANTISSA OF THE DETERMINANT OF THE AUGMENTED JACOBIANPTCN0574
C  IEXP    = BINARY EXPONENT OF THE DETERMINANT OF THE AUGMENTED JACOBIANPTCN0575
C  CURVCF = ESTIMATED CURVATURE BETWEEN XC AND XF.                         PTCN0576
C  CORDIS = NORM OF THE CORRECTOR STEP FROM PREDICTED POINT TO CORRECTEDPTCN0577
C           POINT, USING MAXIMUM ABSOLUTE VALUE AS THE NORM.               PTCN0578
C           THIS QUANTITY IS MODIFIED TO AN 'OPTIMAL' VALUE.               PTCN0579
C  ALPHLC = ANGLE BETWEEN OLD AND NEW TANGENTS TL AND TC                   PTCN0580
C  HSECLC = EUCLIDEAN NORM OF SECANT BETWEEN XL AND XC.                    PTCN0581
C  FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT NEW CONTINUATION POINT.      PTCN0582
C                                                                          PTCN0583
C  /TOL/                                                                   PTCN0584
C  EPMACH= SMALLEST NUMBER SUCH THAT 1.0+EPMACH.GT.EPMACH                  PTCN0585
C          .5*BETA**(1-TAU) FOR ROUNDED, TAU-DIGIT ARITHMETIC              PTCN0586
C          BASE BETA.  TWICE THIS VALUE FOR TRUNCATED ARITHMETIC.          PTCN0587
C          THIS IS THE RELATIVE MACHINE PRECISION.                         PTCN0588
C          EPMACH=2**(-27) FOR DEC-10.                                     PTCN0589
C  EPSATE= 8*EPMACH                                                        PTCN0590
C  EPSQRT= SQUARE ROOT OF EPMACH                                           PTCN0591
C                                                                          PTCN0592
C                                                                          PTCN0593
C  PROGRAMMING NOTES                                                       PTCN0594
C                                                                          PTCN0595
C                                                                          PTCN0596
C  THE USER MUST -                                                         PTCN0597
C                                                                          PTCN0598
C  1.  WRITE SURROUTINES                                                   PTCN0599
C  SUPPLY A CALLING PROGRAM, AND THE TWO ROUTINES FCTN AND FPRIME          PTCN0600
C  AS DESCRIBED ABOVE.                                                     PTCN0601
C                                                                          PTCN0602
C  2.  SET STORAGE AREAS                                                   PTCN0603
C  DECLARE A REAL VECTOR RWORK OF SIZE ISIZE, ISIZE.GE.NVAR*(NVAR+5)       PTCN0604
C  AND AN INTEGER VECTOR IPVT OF SIZE NVAR.                                PTCN0605
C                                                                          PTCN0606
C  3.  PASS CERTAIN NON-DEFAULTABLE VALUES                                 PTCN0607
C  PASS NVAR GREATER THAN ZERO, ISIZE.GE.NVAR*(NVAR+5)                     PTCN0608
C  AND SET IRET=0, KSTEP=-1 OR KSTEP=0 ON FIRST CALL                       PTCN0609
C  FOR A NEW PROBLEM.                                                      PTCN0610
C                                                                          PTCN0611
C  THE USER SHOULD -                                                       PTCN0612
C                                                                          PTCN0613
C  1.  STORE A STARTING POINT XR IN THE FIRST NVAR LOCATIONS OF RWORK       PTCN0614
C  BEFORE CALLING PITCON.                                                  PTCN0615
C  IF SUCH A VALUE IS NOT GIVEN, THE CODE MAY BE UNABLE TO PRODUCE ONE.    PTCN0616
C                                                                          PTCN0617
C  2.  CAREFULLY MONITOR THE VALUE OF IRET SO THAT ANY SERIOUS ERROR       PTCN0618
C  IS CAUGHT BEFORE ANOTHER CALL IS MADE TO PITCON.                        PTCN0619
C                                                                          PTCN0620
C  3.  CHOOSE A VALUE OF IMOD FOR THE TYPE OF CORRECTOR PROCESS TO         PTCN0621
C  BE USED.                                                                PTCN0622
C                                                                          PTCN0623
C                                                                          PTCN0624
C  THE USER MAY -                                                          PTCN0625
C                                                                          PTCN0626
C  1.  MONITOR THE PASSING OF BIFURCATION POINTS BY SAVING THE OLD         PTCN0627
C  VALUE OF DETA AND COMPARING IT TO THE CURRENT VALUE.  IF THERE          PTCN0628
C  IS A CHANGE IN SIGN, THEN A BIFURCATION POINT HAS BEEN PASSED.          PTCN0629
C                                                                          PTCN0630
```

```
C
C  2.  ACCESS THE COMMON BLOCKS /COUNT1/ AND /COUNT2/ TO KEEP TRACK      PTCN0631
C  OF THE AMOUNT OF WORK DONE.                                           PTCN0632
C                                                                        PTCN0633
C  3.  MONITOR THE COMMON BLOCK /POINT/ FOR INFORMATION ABOUT THE        PTCN0634
C  SOLUTION CURVE.                                                       PTCN0635
C                                                                        PTCN0636
C  4.  AT ANY TIME, RESET THE CODE BY PASSING IN KSTEP=-1 OR KSTEP=0.    PTCN0637
C  THIS ALLOWS THE USER TO CHANGE STEPSIZE, DIRECTION OF CONTINUATION,   PTCN0638
C  ERROR CONTROLS, OR OTHER PARAMETERS.  IT ALSO ENABLES THE USER        PTCN0639
C  TO RUN UNRELATED PROBLEMS OF DIFFERENT SIZES OR ERROR CONTROLS        PTCN0640
C  DURING A SINGLE PROGRAM EXECUTION.                                    PTCN0641
C                                                                        PTCN0642
C                                                                        PTCN0643
C  THIS SUBROUTINE IS CALLED BY                                          PTCN0644
C    USER MAIN PROGRAM                                                   PTCN0645
C  AND CALLS                                                             PTCN0646
C    CORECT                                                              PTCN0647
C    ROOT                                                                PTCN0648
C    TANGNT                                                              PTCN0649
C    FORTRAN ABS                                                         PTCN0650
C    FORTRAN ACOS                                                        PTCN0651
C    FORTRAN ALOG                                                        PTCN0652
C    FORTRAN AMAX1                                                       PTCN0653
C    FORTRAN AMIN1                                                       PTCN0654
C    FORTRAN DBLE                                                        PTCN0655
C    FORTRAN FLOAT                                                       PTCN0656
C    FORTRAN SIGN                                                        PTCN0657
C    FORTRAN SIN                                                         PTCN0658
C    FORTRAN SNGL                                                        PTCN0659
C    FORTRAN SQRT                                                        PTCN0660
C    LINPAK ISAMAX                                                       PTCN0661
C    LINPAK SAXPY                                                        PTCN0662
C    LINPAK SCOPY                                                        PTCN0663
C    LINPAK SNRM2                                                        PTCN0664
C    LINPAK SSCAL                                                        PTCN0665
C                                                                        PTCN0666
C************************************************************************PTCN0667
C                                                                        PTCN0668
      INTEGER IPVT(NVAR)                                                 PTCN0669
      REAL RWORK(ISIZE)                                                  PTCN0670
      REAL WRGE(8),ACOF(12)                                              PTCN0671
      DOUBLE PRECISION DTLIPC,DTCIPC,DADJUS,COSALF                       PTCN0672
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT          PTCN0673
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM          PTCN0674
      COMMON /OUTPUT/ IWRITE                                             PTCN0675
      COMMON /POINT/  DETA,IEXP,CURVCF,CORDIS,ALPHLC,HSECLC,FNRMXF       PTCN0676
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT                               PTCN0677
      DATA IDONE  /0/                                                    PTCN0678
      DATA TENM1  /0.1/                                                  PTCN0679
      DATA TENM2  /0.01/                                                 PTCN0680
      DATA TENM3  /0.001/                                                PTCN0681
      DATA WRGE   /                                                      PTCN0682
     1 .8735115E+00, .1531947E+00, .3191815E-01, .3339946E-10,           PTCN0683
     2 .4677788E+00, .6970123E-03, .1980863E-05, .1122789E-08/           PTCN0684
      DATA ACOF   /                                                      PTCN0685
     1 .9043128E+00,-.7075675E+00,-.4667383E+01,-.3677482E+01,           PTCN0686
     2 .8516099E+00,-.1953119E+00,-.4830636E+01,-.9770528E+00,           PTCN0687
     3 .1040061E+01, .3793395E-01, .1042177E+01, .4450706E-01/           PTCN0688
C                                                                        PTCN0689
C************************************************************************PTCN0690
C                                                                        PTCN0691
C  1.  PREPARATIONS.                                                     PTCN0692
C  ON FIRST CALL FOR THIS PROBLEM, INITIALIZE COUNTERS AND VARIABLES,    PTCN0693
C  CHECK USER INFORMATION AND SET DEFAULTS, AND IF (KSTEP.EQ.-1),        PTCN0694
C  CHECK NORM OF F(XR) AND CORRECT XR IF NECESSARY.                      PTCN0695
C  ON EACH CALL, IF INPUT IRET HAS NONFATAL VALUE, RESET IRET            PTCN0696
C  SO THAT CONTINUATION LOOP PICKS UP WHERE IT WAS HALTED.               PTCN0697
C                                                                        PTCN0698
C************************************************************************PTCN0699
C                                                                        PTCN0700
```

```
      IERR=0                                                            PTCN0701
      IF (IRET.EQ.-1) IRET=2                                            PTCN0702
      IF (IRET.EQ.-2.OR.IRET.EQ.-3.OR.IRET.EQ.-4) IRET=1                PTCN0703
      IF (IRET.EQ.-5.OR.IRET.EQ.-6) IRET=0                              PTCN0704
C                                                                       PTCN0705
C  IF CODE WAS CALLED AGAIN AFTER FATAL VALUE OF IRET,                  PTCN0706
C  THEN RETURN WITH ERROR VALUE IRET=-10.                               PTCN0707
C                                                                       PTCN0708
      IF (IRET.LT.0) GO TO 440                                          PTCN0709
C                                                                       PTCN0710
C  PERFORM ONE-TIME ONLY INITIALIZATIONS                                PTCN0711
C                                                                       PTCN0712
      IF (IDONE.NE.0) GO TO 10                                          PTCN0713
C                                                                       PTCN0714
C  SET THE MACHINE DEPENDENT VARIABLE EPMACH, THE SMALLEST NUMBER       PTCN0715
C  SO THAT (1.0+EPMACH.GT.1.0)                                          PTCN0716
C                                                                       PTCN0717
C  FOR DEC PDP-10 IN SINGLE PRECISION:                                  PTCN0718
C                                                                       PTCN0719
      EPMACH=7.4505806E-9                                               PTCN0720
C                                                                       PTCN0721
C  FOR IBM 360 OR 370 IN SHORT (SINGLE) PRECISION:                      PTCN0722
C                                                                       PTCN0723
C     EPMACH=9.53674E-7                                                PTCN0724
C                                                                       PTCN0725
C  FOR CDC 6600 OR 7400 IN SINGLE PRECISION:                            PTCN0726
C                                                                       PTCN0727
C     EPMACH=7.105427406E-15                                           PTCN0728
C                                                                       PTCN0729
C  SET EPSATE=8*EPMACH, EPSQRT=SQRT(EPMACH)                             PTCN0730
C                                                                       PTCN0731
      EPSATE=8.0*EPMACH                                                 PTCN0732
      EPSQRT=SQRT(EPMACH)                                               PTCN0733
      ALFMIN=2.0*ACOS(1.0-EPMACH)                                       PTCN0734
      IF(KSTEP.LT.-1.OR.KSTEP.GT.0)KSTEP=-1                             PTCN0735
      KSTEPO=-2                                                         PTCN0736
      IDONE=1                                                           PTCN0737
C                                                                       PTCN0738
C  PERFORM INITIALIZATIONS AND CHECKS FOR NEW PROBLEM ONLY              PTCN0739
C                                                                       PTCN0740
   10 IF (KSTEP.GT.0) GO TO 30                                          PTCN0741
      IF(KSTEPO.EQ.-1.AND.KSTEP.EQ.0)GO TO 30                           PTCN0742
      IF (NVAR.LE.1) GO TO 440                                          PTCN0743
      IF (ISIZE.LT.(NVAR)*(NVAR+5)) GO TO 440                           PTCN0744
      IXR=1                                                             PTCN0745
      JXR=0                                                             PTCN0746
      IXC=IXR+NVAR                                                      PTCN0747
      JXC=JXR+NVAR                                                      PTCN0748
      IXF=IXC+NVAR                                                      PTCN0749
      JXF=JXC+NVAR                                                      PTCN0750
      ITL=IXF+NVAR                                                      PTCN0751
      JTL=JXF+NVAR                                                      PTCN0752
      ITC=ITL+NVAR                                                      PTCN0753
      JTC=JTL+NVAR                                                      PTCN0754
      IFP=ITC+NVAR                                                      PTCN0755
      JFP=JTC+NVAR                                                      PTCN0756
      BETA=0.0                                                          PTCN0757
      TCIPC=0.0                                                         PTCN0758
      CORDIS=0.0                                                        PTCN0759
      CURVCF=0.0                                                        PTCN0760
      HSECLL=0.0                                                        PTCN0761
      HSECLC=0.0                                                        PTCN0762
      XITO=0.0                                                          PTCN0763
      ITO=0                                                             PTCN0764
      NEQN=NVAR-1                                                       PTCN0765
      NRED=0                                                            PTCN0766
      KNSUM=0                                                           PTCN0767
      NRDSUM=0                                                          PTCN0768
      ICRSL=0                                                           PTCN0769
      ITNSL=0                                                           PTCN0770
```

```
      NSTCR=0                                                          PTCN0771
      NCNCR=0                                                          PTCN0772
      NTRCR=0                                                          PTCN0773
      NLMCR=0                                                          PTCN0774
      NLMRT=0                                                          PTCN0775
      IFEVAL=0                                                         PTCN0776
      ISOLVE=0                                                         PTCN0777
      IPEVAL=0                                                         PTCN0778
      IF (HMAX.LE.0.0) HMAX=SQRT(FLOAT(NVAR))                          PTCN0779
      IF (HMIN.LE.EPSQRT) HMIN=EPSQRT                                  PTCN0780
      HDEF=.5*(HMAX+HMIN)                                              PTCN0781
      IF (HFACT.LE.1.0) HFACT=3.0                                      PTCN0782
      HRED=1.0/HFACT                                                   PTCN0783
      IF (ABSERR.LE.0.0) ABSERR=EPSQRT                                 PTCN0784
      IF (RELERR.LE.0.0) RELERR=EPSQRT                                 PTCN0785
      IF (IPC.LE.0.OR.IPC.GT.NVAR) IPC=NVAR                            PTCN0786
      IF (LIM.LT.0.OR.LIM.GT.NVAR) LIM=0                               PTCN0787
      IF (H.EQ.0.0) H=HDEF                                             PTCN0788
      DIR=SIGN(1.0,H)                                                  PTCN0789
      H=ABS(H)                                                         PTCN0790
C                                                                      PTCN0791
C  IF (KSTEP.LT.0) CHECK NORM OF F(XR) AT STARTING POINT,              PTCN0792
C  IF ACCEPTABLE, RETURN IMMEDIATELY WITH KSTEP=0,                     PTCN0793
C  OTHERWISE APPLY NEWTON'S METHOD, HOLDING VALUE OF                   PTCN0794
C  IPC-TH COMPONENT FIXED.                                             PTCN0795
C                                                                      PTCN0796
      IF (KSTEP.GE.0) GO TO 20                                         PTCN0797
      CALL CORECT(NVAR,RWORK(IXR),IPC,RWORK(ITL),IERR,IMOD,RWORK(IFP), PTCN0798
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRMXF)                           PTCN0799
      NSTCR=NSTCR+KN                                                   PTCN0800
C                                                                      PTCN0801
C  IF NO ACCEPTABLE POINT FOUND, ERROR RETURN                          PTCN0802
C                                                                      PTCN0803
      IF (IERR.NE.0) GO TO 400                                         PTCN0804
      KSTEP0=-1                                                        PTCN0805
      KSTEP=0                                                          PTCN0806
      HTANCF=H                                                         PTCN0807
      GO TO 340                                                        PTCN0808
   20 IF (KSTEP.EQ.0) CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXC),1)       PTCN0809
      IF (KSTEP.EQ.0) CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXF),1)       PTCN0810
C                                                                      PTCN0811
C***********************************************************************PTCN0812
C                                                                      PTCN0813
C  2.  TARGET POINT CHECK.  IF (IT.NE.0) TARGET POINTS ARE SOUGHT.     PTCN0814
C  CHECK TO SEE IF TARGET COMPONENT IT HAS VALUE XIT LYING             PTCN0815
C  BETWEEN XC(IT) AND XF(IT).  IF SO, GET LINEARLY INTERPOLATED        PTCN0816
C  STARTING POINT, AND USE NEWTON'S METHOD TO GET TARGET POINT         PTCN0817
C                                                                      PTCN0818
C***********************************************************************PTCN0819
C                                                                      PTCN0820
   30 IF(IT.LT.0.OR.IT.GT.NVAR)IT=0                                    PTCN0821
      IF (IT.EQ.0) GO TO 40                                            PTCN0822
      IF (IRET.EQ.1.AND.XIT.EQ.XITO.AND.IT.EQ.ITO) GO TO 40            PTCN0823
      XCIT=RWORK(JXC+IT)                                               PTCN0824
      XFIT=RWORK(JXF+IT)                                               PTCN0825
      IF ((XIT.LT.XCIT).AND.(XIT.LT.XFIT)) GO TO 40                    PTCN0826
      IF ((XIT.GT.XCIT).AND.(XIT.GT.XFIT)) GO TO 40                    PTCN0827
      DEL=XFIT-XCIT                                                    PTCN0828
      RAT=0.0                                                          PTCN0829
      IF (ABS(DEL).GT.EPSQRT) RAT=(XIT-XCIT)/DEL                       PTCN0830
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                       PTCN0831
      CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)                  PTCN0832
      CALL SSCAL(NVAR,RAT,RWORK(IXR),1)                                PTCN0833
      CALL SAXPY(NVAR,1.0,RWORK(IXC),1,RWORK(IXR),1)                   PTCN0834
      RWORK(JXR+IT)=XIT                                                PTCN0835
      CALL CORECT(NVAR,RWORK(IXR),IT,RWORK(ITL),IERR,IMOD,RWORK(IFP),  PTCN0836
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRM)                             PTCN0837
      NTRCR=NTRCR+KN                                                   PTCN0838
      ITO=IT                                                           PTCN0839
      XITO=XIT                                                         PTCN0840
```

```
      IF (IERR.EQ.0) GO TO 320                                          PTCN0841
      IF (IERR.EQ.-1) GO TO 370                                         PTCN0842
      IF (IERR.EQ.-2) GO TO 360                                         PTCN0843
      IF (IERR.EQ.-3) GO TO 380                                         PTCN0844
C                                                                       PTCN0845
C*************************************************************************PTCN0846
C                                                                       PTCN0847
C  3.  TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION.  UNLESS THEPTCN0848
C  TANGENT AND LIMIT POINT CALCULATIONS WERE ALREADY PERFORMED (BECAUSE PTCN0849
C  THE LOOP WAS INTERRUPTED FOR A LIMIT POINT), SET UP AND SOLVE       PTCN0850
C  THE EQUATION FOR THE TANGENT VECTOR.  FORCE THE TANGENT VECTOR TO BE PTCN0851
C  OF UNIT LENGTH, AND FORCE THE IPL-COMPONENT TO HAVE THE SAME SIGN AS PTCN0852
C  THE IPL-TH COMPONENT OF THE PREVIOUS TANGENT VECTOR, OR (ON FIRST    PTCN0853
C  STEP) THE SAME SIGN AS THE USER INPUT DIRECTION DIR.  SET THE LOCAL  PTCN0854
C  PARAMETER IPC TO THE LOCATION OF THE LARGEST COMPONENT              PTCN0855
C  OF THE TANGENT VECTOR, UNLESS A LIMIT POINT IN THAT DIRECTION       PTCN0856
C  APPEARS TO BE APPROACHING AND ANOTHER CHOICE IS AVAILABLE.          PTCN0857
C                                                                       PTCN0858
C*************************************************************************PTCN0859
C                                                                       PTCN0860
   40 IF (IRET.NE.2) GO TO 50                                           PTCN0861
      IRET=0                                                            PTCN0862
      GO TO 160                                                         PTCN0863
C                                                                       PTCN0864
C  STORE OLD TANGENT IN TL, COMPUTE NEW TANGENT FOR XC                  PTCN0865
C                                                                       PTCN0866
   50 IPL=IPC                                                           PTCN0867
      IF(KSTEP.GT.0)CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1)          PTCN0868
      ICALL=1                                                           PTCN0869
      CALL TANGNT(NVAR,RWORK(IXF),IPC,RWORK(ITC),IRET,ICALL,RWORK(IFP), PTCN0870
     1 IPVT,NEQN,DETA,IEXP)                                             PTCN0871
      IF (IRET.EQ.-2) GO TO 430                                         PTCN0872
      IF (IRET.EQ.-1) GO TO 410                                         PTCN0873
C                                                                       PTCN0874
C  SUBROUTINE TANGNT RETURNED IPC, THE LOCATION OF THE LARGEST COMPONENTPTCN0875
C  OF THE TANGENT VECTOR.  THIS WILL BE USED FOR THE LOCAL             PTCN0876
C  PARAMETERIZATION OF THE CURVE UNLESS A LIMIT POINT IN IPC SEEMS     PTCN0877
C  TO BE COMING.  TO CHECK THIS, WE COMPARE TCIPC:=TC(IPC) AND THE     PTCN0878
C  SECOND LARGEST COMPONENT TCJPC:=TC(JPC).  IF TCJPC IS NO LESS       PTCN0879
C  THAN .1 OF TCIPC, AND TC(JPC) IS LARGER THAN TL(JPC),               PTCN0880
C  WHEREAS TC(IPC) IS LESS THAN TL(IPC), WE WILL RESET THE             PTCN0881
C  LOCAL PARAMETER IPC:=JPC.                                           PTCN0882
C                                                                       PTCN0883
      TLIPL=TCIPC                                                       PTCN0884
      TCIPC=RWORK(JTC+IPC)                                              PTCN0885
      JPC=IPC                                                           PTCN0886
      IF(ABS(TCIPC).GE.ABS(RWORK(JTL+IPC)))GO TO 60                     PTCN0887
      IF(TLIPL.EQ.0.0)GOTO 60                                           PTCN0888
      RWORK(JTC+IPC)=0.0                                                PTCN0889
      JPC=ISAMAX(NVAR,RWORK(ITC),1)                                     PTCN0890
      TCJPC=RWORK(JTC+JPC)                                              PTCN0891
      RWORK(JTC+IPC)=TCIPC                                              PTCN0892
      IF (ABS(TCJPC).LT.TENM1*ABS(TCIPC)) GO TO 60                      PTCN0893
      IF (ABS(TCJPC).LT.ABS(RWORK(JTL+JPC)))GOTO 60                     PTCN0894
      IPC=JPC                                                           PTCN0895
      IF (IWRITE.GE.2)WRITE(6,610)                                      PTCN0896
   60 TCIPL=RWORK(JTC+IPL)                                              PTCN0897
      DTLIPC=DBLE(RWORK(JTL+IPC))                                       PTCN0898
      DETA=DETA/TCIPL                                                   PTCN0899
C                                                                       PTCN0900
C  ADJUST SIGN OF TANGENT                                               PTCN0901
C  COMPARE THE SIGN OF TC(IPL) WITH SIGN OF TL(IPL)                     PTCN0902
C  (BUT ON FIRST STEP, COMPARE SIGN OF TC(IPL) WITH USER INPUT DIR).    PTCN0903
C  IF THESE SIGNS DIFFER, CHANGE THE SIGN OF TC TO FORCE AGREEMENT,     PTCN0904
C  AND THE SIGN OF DETA.                                               PTCN0905
C  THEN RECORD DIR:= SIGN OF DETERMINANT = SIGN(DETA).                  PTCN0906
C                                                                       PTCN0907
      STLIPL=DIR                                                        PTCN0908
      IF (TLIPL.NE.0.0) STLIPL=SIGN(1.0,TLIPL)                          PTCN0909
      IF (SIGN(1.0,TCIPL).EQ.STLIPL) GO TO 70                           PTCN0910
```

```
      CALL SSCAL(NVAR,-1.0,RWORK(ITC),1)                                PTCN0911
      DETA=-DETA                                                        PTCN0912
      TCIPL=-TCIPL                                                      PTCN0913
   70 TCIPC=RWORK(JTC+IPC)                                              PTCN0914
      TCJPC=RWORK(JTC+JPC)                                              PTCN0915
      DTCIPC=DBLE(TCIPC)                                                PTCN0916
      DIR=SIGN(1.0,DETA)                                                PTCN0917
      IF (LIM.EQ.0) GO TO 80                                            PTCN0918
      TLLIM=TCLIM                                                       PTCN0919
      TCLIM=RWORK(JTC+LIM)                                              PTCN0920
C                                                                       PTCN0921
C  COMPUTE ALPHLC, THE ANGLE BETWEEN TANGENT AT XL AND TANGENT AT XC    PTCN0922
C  AND HSECLC, THE EUCLIDEAN NORM OF SECANT FROM XL TO XC.              PTCN0923
C                                                                       PTCN0924
   80 IF(KSTEP.LE.0) GO TO 160                                          PTCN0925
      COSALF=0.000                                                      PTCN0926
      DO 90 I=1,NVAR                                                    PTCN0927
           COSALF=COSALF+DBLE(RWORK(JTC+I))*DBLE(RWORK(JTL+I))          PTCN0928
   90      RWORK(JXR+I)=RWORK(JXF+I)-RWORK(JXC+I)                       PTCN0929
      HSECLL=HSECLC                                                     PTCN0930
      HSECLC=SNRM2(NVAR,RWORK(IXR),1)                                   PTCN0931
      ALPHLC=SNGL(COSALF)                                               PTCN0932
      IF(ALPHLC.GT.1.0)ALPHLC=1.0                                       PTCN0933
      IF(ALPHLC.LT.-1.0)ALPHLC=-1.0                                     PTCN0934
      ALPHLC=ACOS(ALPHLC)                                               PTCN0935
      IF(IWRITE.GE.2)WRITE(6,550)ALPHLC                                 PTCN0936
C                                                                       PTCN0937
C***********************************************************************PTCN0938
C                                                                       PTCN0939
C  4.  LIMIT POINT CHECK.  IF (LIM.NE.0) CHECK TO SEE IF                PTCN0940
C  OLD AND NEW TANGENTS DIFFER IN SIGN OF LIM-TH COMPONENT.             PTCN0941
C  IF SO, ATTEMPT TO COMPUTE A POINT XR BETWEEN XC AND XF               PTCN0942
C  FOR WHICH TANGENT COMPONENT VANISHES                                 PTCN0943
C                                                                       PTCN0944
C***********************************************************************PTCN0945
C                                                                       PTCN0946
      IF (LIM.LE.0.OR.KSTEP.LE.0) GO TO 160                             PTCN0947
C                                                                       PTCN0948
C  CHECK FOR LIMIT INTERVAL                                             PTCN0949
C                                                                       PTCN0950
      IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160                 PTCN0951
C                                                                       PTCN0952
C  TEST FOR ACCEPTABLE ENDPOINTS                                        PTCN0953
C                                                                       PTCN0954
      ATLLM=ABS(TLLIM)                                                  PTCN0955
      IF (ATLLM.GT.0.5*ABSERR) GO TO 110                                PTCN0956
C                                                                       PTCN0957
C  IF XC IS LIMIT POINT, TL ALREADY CONTAINS TANGENT AT XC              PTCN0958
C                                                                       PTCN0959
  100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1)                        PTCN0960
      GO TO 310                                                         PTCN0961
  110 ATCLIM=ABS(TCLIM)                                                 PTCN0962
      IF (ATCLIM.GT.0.5*ABSERR) GO TO 130                               PTCN0963
  120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                        PTCN0964
      CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1)                        PTCN0965
      GO TO 310                                                         PTCN0966
C                                                                       PTCN0967
C  TEST FOR SMALL INTERVAL                                              PTCN0968
C                                                                       PTCN0969
  130 XCLIM=RWORK(IXC+LIM)                                              PTCN0970
      XFLIM=RWORK(JXF+LIM)                                              PTCN0971
      DEL=ABS(XFLIM-XCLIM)                                              PTCN0972
      XABS=AMAX1(ABS(XCLIM),ABS(XFLIM))                                 PTCN0973
      IF (DEL.GT.(ABSERR+RELERR*XABS)) GO TO 140                        PTCN0974
      IF (ATLLM.GT.ATCLIM) GO TO 120                                    PTCN0975
      GO TO 100                                                         PTCN0976
C                                                                       PTCN0977
C  BEGIN ROOT-FINDING ITERATION WITH INTERVAL (0,1) AND                 PTCN0978
C  FUNCTION VALUES TL(LIM), TC(LIM).                                    PTCN0979
C                                                                       PTCN0980
```

```
  140 KOUNT=0                                                                    PTCN0981
      A=0.0                                                                      PTCN0982
      FA=TLLIM                                                                   PTCN0983
      TSN=TCLIM                                                                  PTCN0984
      C=1.0                                                                      PTCN0985
      FC=TCLIM                                                                   PTCN0986
C                                                                                PTCN0987
C  SET IPLIM TO THE INDEX OF MAXIMUM ENTRY OF SECANT                             PTCN0988
C  (EXCEPT THAT IPLIM MUST NOT EQUAL LIM)                                        PTCN0989
C  AND SAVE THE SIGN OF THE MAXIMUM COMPONENT IN DIRLIM                          PTCN0990
C  SO THAT NEW TANGENTS MAY BE PROPERLY SIGNED.                                  PTCN0991
C                                                                                PTCN0992
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                                 PTCN0993
      CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)                            PTCN0994
      RWORK(JXR+LIM)=0.0                                                         PTCN0995
      IPLIM=ISAMAX(NVAR,RWORK(IXR),1)                                            PTCN0996
      DIRLIM=SIGN(1.0,RWORK(JXR+IPLIM))                                          PTCN0997
C                                                                                PTCN0998
C  CALL ROOTFINDER FOR APPROXIMATE ROOT SN, SET X=SN*XF+(1-SN)*XC                PTCN0999
C  CALL CORRECTOR TO RETURN TO CURVE ON LINE X(IPLIM)=CONSTANT,                  PTCN1000
C  COMPUTE TANGENT THERE, AND SET FUNCTION VALUE TO TANGENT(LIM)                 PTCN1001
C                                                                                PTCN1002
  150 CALL ROOT(A,FA,SN,TSN,C,FC,KOUNT,IFLAG)                                    PTCN1003
      NLMRT=NLMRT+1                                                              PTCN1004
      IF((IFLAG.LT.-1)GO TO 350                                                  PTCN1005
      IF(IFLAG.EQ.-1.OR.IFLAG.EQ.0)GO TO 310                                     PTCN1006
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                                 PTCN1007
      CALL SSCAL(NVAR,SN,RWORK(IXR),1)                                           PTCN1008
      SCALER=1.0-SN                                                              PTCN1009
      CALL SAXPY(NVAR,SCALER,RWORK(IXC),1,RWORK(IXR),1)                          PTCN1010
      CALL CORECT(NVAR,RWORK(IXR),IPLIM,RWORK(ITL),IERR,IHOO,RWORK(IFP)          PTCN1011
     1 ,IPVT,ABSERR,RELERR,XSTPLM,NEQN,FNRM)                                     PTCN1012
      NLMCR=NLMCR+KN                                                             PTCN1013
      IF (IERR.NE.0) GO TO 350                                                   PTCN1014
      ICALL=1                                                                    PTCN1015
      IPT=IPLIM                                                                  PTCN1016
      CALL TANGNT(NVAR,RWORK(IXR),IPT,RWORK(ITL),IRET,ICALL,                     PTCN1017
     1 RWORK(IFP),IPVT,NEQN,DETLIM,IEXLIM)                                       PTCN1018
      IF(IRET.LT.0)GO TO 350                                                     PTCN1019
C                                                                                PTCN1020
C  ADJUST THE SIGN OF THE TANGENT SO THAT THE IPLIM-TH COMPONENT                 PTCN1021
C  HAS THE SAME SIGN AS THE IPLIM-TH COMPONENT OF THE SECANT                     PTCN1022
C                                                                                PTCN1023
      IF (DIRLIM.NE.SIGN(1.0,RWORK(JTL+IPLIM)))                                  PTCN1024
     1 CALL SSCAL(NVAR,-1.0,RWORK(ITL),1)                                        PTCN1025
C                                                                                PTCN1026
C  SEE IF WE CAN ACCEPT THE NEW POINT BECAUSE TANGENT(LIM) IS SMALL              PTCN1027
C  OR MUST 'ACCEPT' THE POINT BECAUSE THE VALUES ARE NOT DECREASING              PTCN1028
C  RAPIDLY ENOUGH, OR IF WE CAN GO ON.                                           PTCN1029
C                                                                                PTCN1030
      TSNOLD=TSN                                                                 PTCN1031
      TSN=RWORK(JTL+LIM)                                                         PTCN1032
      IF (ABS(TSN).LE.0.5*ABSERR) GO TO 310                                      PTCN1033
      GO TO 150                                                                  PTCN1034
C                                                                                PTCN1035
C********************************************************************************PTCN1036
C                                                                                PTCN1037
C  5.  STEP LENGTH COMPUTATION.  COMPUTE STEPLENGTH HTANCF                       PTCN1038
C                                                                                PTCN1039
C  THE FORMULAS UNDERLYING THE ALGORITHM ARE                                     PTCN1040
C                                                                                PTCN1041
C  LET                                                                           PTCN1042
C                                                                                PTCN1043
C  ALPHLC = MAXIMUM OF ARCCOS(TL,TC) AND ALFMIN = 2*ARCCOS(1-EPMACH)             PTCN1044
C  HSECLC = NORM(XL-XC)                                                          PTCN1045
C  HSECLL = NORM(XL-XLL)                                                         PTCN1046
C  ABSIN  = ABS(SIN(.5*ALPHLC))                                                  PTCN1047
C  CURVLC = LAST VALUE OF CURVCF                                                 PTCN1048
C  CURVCF = 2*ABSIN/HSECLC                                                       PTCN1049
C  CORDIS = OPTIMIZED CORRECTOR DISTANCE TO CURRENT CONTINUATION POINT.          PTCN1050
```

```
C           BUT CORDIS FORCED TO LIE BETWEEN .1*HSECLC AND HSECLC.      PTCN1051
C           UNLESS (CORDIS.EQ.0.0), BECAUSE THE PREDICTED POINT WAS     PTCN1052
C           IMMEDIATELY ACCEPTED.  IN SUCH A CASE, SET HTANCF=HSECLC    PTCN1053
C           INSTEAD OF USING FIRST ESTIMATE FOR HTANCF.                 PTCN1054
C                                                                       PTCN1055
C  THEN                                                                 PTCN1056
C                                                                       PTCN1057
C  CURVXF = CURVCF + HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL)             PTCN1058
C  BUT CURVXF MUST BE GREATER THAN .001, AND A SIMPLER FORMULA IS USED  PTCN1059
C  IF WE DO NOT HAVE DATA AT TWO PREVIOUS POINTS.                       PTCN1060
C                                                                       PTCN1061
C  FIRST ESTIMATE:  (UNLESS (CORDIS.EQ.0.0) )                           PTCN1062
C                                                                       PTCN1063
C  HTANCF   = SQRT(2*CORDIS/CURVXF)                                     PTCN1064
C                                                                       PTCN1065
C  ADJUSTED VALUE:                                                      PTCN1066
C                                                                       PTCN1067
C  HTANCF = HTANCF*(1.0 + HTANCF*(TC(IPC)-TL(IPC))/(2*HSECLC*TC(IPC)))  PTCN1068
C                                                                       PTCN1069
C  READJUSTMENT AND TRUNCATIONS:                                        PTCN1070
C                                                                       PTCN1071
C  IF STEPSIZE REDUCTION OCCURRED DURING LAST CORRECTOR PROCESS,        PTCN1072
C  HTANCF IS FORCED TO BE LESS THAN (HFACT-1)*HSECLC/2.                 PTCN1073
C                                                                       PTCN1074
C  HTANCF MUST LIE BETWEEN (HSECLC/HFACT) AND (HSECLC*HFACT).           PTCN1075
C                                                                       PTCN1076
C  HTANCF IS ALWAYS FORCED TO LIE BETWEEN HMIN AND HMAX.                PTCN1077
C                                                                       PTCN1078
C***********************************************************************PTCN1079
C                                                                       PTCN1080
C  CHECK IF DEFAULT STEP MUST BE USED:                                  PTCN1081
C  ON FIRST STEP, USE HTANCF=H.                                         PTCN1082
C  IF PREVIOUS STEP WAS OF SIZE ZERO, USE STEPSIZE HDEF=(HMIN+HMAX)/2   PTCN1083
C                                                                       PTCN1084
  160 IF (KSTEP.GT.0.AND.HSECLC.GT.0.0) GO TO 170                       PTCN1085
      HTANCF=HDEF                                                       PTCN1086
      IF(KSTEP.LE.0)HTANCF=H                                            PTCN1087
      GO TO 190                                                         PTCN1088
  170 IF(ALPHLC.LT.ALFMIN)ALPHLC=ALFMIN                                 PTCN1089
      ARSIN=ABS(SIN(.5*ALPHLC))                                         PTCN1090
C                                                                       PTCN1091
C  COMPUTE NEW CURVATURE DATA                                           PTCN1092
C                                                                       PTCN1093
      CURVLC=CURVCF                                                     PTCN1094
      CURVCF=2.0*ARSIN/HSECLC                                           PTCN1095
      CURVXF=CURVCF                                                     PTCN1096
      IF(HSECLL.NE.0.0)                                                 PTCN1097
     1 CURVXF=CURVCF+HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL)             PTCN1098
      CURVXF=AMAX1(CURVXF,TENM3)                                        PTCN1099
      IF(IWRITE.GE.2)WRITE(6,560)CURVCF,CURVXF                          PTCN1100
C                                                                       PTCN1101
C  IF THE CONVERGENCE DISTANCE IS ZERO, SET FIRST ESTIMATE TO HSECLC.   PTCN1102
C  OTHERWISE, TRUNCATE CORDIS TO LIE BETWEEN .01*HSECLC AND HSECLC.     PTCN1103
C                                                                       PTCN1104
      HTANCF=HSECLC                                                     PTCN1105
      IF(CORDIS.EQ.0.0) GO TO 180                                       PTCN1106
      TEMP=TENM2*HSECLC                                                 PTCN1107
      CORDIS=AMAX1(CORDIS,TEMP)                                         PTCN1108
      CORDIS=AMIN1(CORDIS,HSECLC)                                       PTCN1109
C                                                                       PTCN1110
C  SET HTANCF, THEN ADJUST FOR CURVATURE IN CONTINUATION                PTCN1111
C  PARAMETER DIRECTION, THEN TRUNCATE                                   PTCN1112
C                                                                       PTCN1113
      HTANCF=SQRT(2.0*CORDIS/CURVXF)                                    PTCN1114
  180 IF(NRED.GT.0)HTANCF=AMIN1(HTANCF,(HFACT-1.0)*HSECLC*.5)           PTCN1115
      DADJUS=1.0D0+(1.0D0-DTLIPC/DTCIPC)*DBLE(.5*HTANCF)/DBLE(HSECLC)   PTCN1116
      HTANCF=HTANCF*SNGL(DADJUS)                                        PTCN1117
      TEMP=HSECLC*HRED                                                  PTCN1118
      HTANCF=AMAX1(HTANCF,TEMP)                                         PTCN1119
      TEMP=HSECLC*HFACT                                                 PTCN1120
```

```
      HTANCF=AMIN1(HTANCF,TEMP)                                          PTCN1121
      HTANCF=AMAX1(HTANCF,HMIN)                                          PTCN1122
      HTANCF=AMIN1(HTANCF,HMAX)                                          PTCN1123
C                                                                        PTCN1124
C*************************************************************************PTCN1125
C                                                                        PTCN1126
C  6.  PREDICTION AND CORRECTION STEPS.  USING XR=XC+HTANCF*TC           PTCN1127
C  AS STARTING POINT, CORRECT XR WITH A FULL OR MODIFIED NEWTON          PTCN1128
C  ITERATION.  IF CORECT FAILS, REDUCE STEPSIZE USED FOR PREDICTOR       PTCN1129
C  POINT, AND TRY AGAIN.  CORRECTION WILL ONLY BE ABANDONED IF STEPSIZE  PTCN1130
C  FALLS BELOW HMIN.                                                     PTCN1131
C                                                                        PTCN1132
C*************************************************************************PTCN1133
C                                                                        PTCN1134
  190 KSTEP0=KSTEP                                                       PTCN1135
      KSTEP=KSTEP+1                                                      PTCN1136
      NRED=0                                                             PTCN1137
  200 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                         PTCN1138
      CALL SAXPY(NVAR,HTANCF,RWORK(ITC),1,RWORK(IXR),1)                  PTCN1139
  210 IF(IWRITE.GE.2)WRITE(6,570)HTANCF                                  PTCN1140
      IF(IWRITE.GE.3)WRITE(6,580)(RWORK(JXR+I),I=1,NVAR)                 PTCN1141
      CALL CORECT(NVAR,RWORK(IXR),IPC,RWORK(ITL),IERR,IMOD,RWORK(IFP),   PTCN1142
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRMXF)                             PTCN1143
      NCNCR=NCNCR+KN                                                     PTCN1144
      IF (IERR.EQ.0) GO TO 230                                           PTCN1145
      IF (IERR.EQ.-1) GO TO 420                                          PTCN1146
C                                                                        PTCN1147
C  NO CONVERGENCE, TRY A SMALLER STEPSIZE                                PTCN1148
C                                                                        PTCN1149
      HTANCF=HRED*HTANCF                                                 PTCN1150
      IF (HTANCF.LT.HMIN) GO TO 390                                      PTCN1151
      NRED=NRED+1                                                        PTCN1152
      IF (IERR.EQ.-2) GO TO 220                                          PTCN1153
      GO TO 200                                                          PTCN1154
  220 CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)                    PTCN1155
      CALL SSCAL(NVAR,HRED,RWORK(IXR),1)                                 PTCN1156
      CALL SAXPY(NVAR,1.0,RWORK(IXC),1,RWORK(IXR),1)                     PTCN1157
      GO TO 210                                                          PTCN1158
C                                                                        PTCN1159
C*************************************************************************PTCN1160
C                                                                        PTCN1161
C  7.  SUCCESSFUL STEP.  STORE INFORMATION BEFORE RETURN.                PTCN1162
C  UPDATE OLD AND CURRENT CONTINUATION POINTS.                           PTCN1163
C  COMPUTE CORDIS, THE SIZE OF THE CORRECTOR STEP.  COMPUTE              PTCN1164
C  A FACTOR THETA WHICH RESCALES CORDIS TO A VALUE WHICH WOULD           PTCN1165
C  CORRESPOND TO A DESIRABLE NUMBER OF CORRECTOR STEPS                   PTCN1166
C  (4 FOR FULL NEWTON, 10 FOR MODIFIED NEWTON).                          PTCN1167
C  SEE REFERENCE DEN HEIJER AND RHEINBOLDT, LOC. CIT.                    PTCN1168
C                                                                        PTCN1169
C*************************************************************************PTCN1170
C                                                                        PTCN1171
  230 NRDSUM=NRDSUM+NRED                                                 PTCN1172
      IF(NRED.NE.0.AND.IWRITE.GE.2)WRITE(6,590)NRED                      PTCN1173
C                                                                        PTCN1174
C  COMPUTE CORRECTOR STEP = XC+HTANCF*TC-XF                              PTCN1175
C  SET CORDIS = MAX NORM OF CORRECTOR STEP                               PTCN1176
C                                                                        PTCN1177
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXC),1)                         PTCN1178
      CALL SAXPY(NVAR,-1.0,RWORK(IXR),1,RWORK(IXC),1)                    PTCN1179
      CALL SAXPY(NVAR,HTANCF,RWORK(ITC),1,RWORK(IXC),1)                  PTCN1180
      IMAX=ISAMAX(NVAR,RWORK(IXC),1)                                     PTCN1181
      CORDIS=ABS(RWORK(JXC+IMAX))                                        PTCN1182
      IF(KN.EQ.0) CORDIS=0.0                                             PTCN1183
C                                                                        PTCN1184
C  MODIFY CORDIS TO A VALUE THAT WOULD CORRESPOND TO THE                 PTCN1185
C  DESIRED NUMBER OF CORRECTOR STEPS                                     PTCN1186
C                                                                        PTCN1187
      IF(CORDIS.EQ.0.0)GO TO 300                                         PTCN1188
      OMEGA=XSTEP/CORDIS                                                 PTCN1189
      THETA=0.0                                                          PTCN1190
```

```
      IF(IMOD.EQ.1) GOTO 260                                          PTCN1191
C                                                                     PTCN1192
C  FULL NEWTON METHOD FOR CORRECTOR STEPS                             PTCN1193
C                                                                     PTCN1194
      IF(KN.LE.1)THETA=8.0                                            PTCN1195
      IF(KN.EQ.4)THETA=1.0                                            PTCN1196
      IF(THETA.NE.0.0)GO TO 290                                       PTCN1197
      IF(KN.GT.4)GO TO 240                                            PTCN1198
      LK=4*KN-7                                                       PTCN1199
      THETA=1.0                                                       PTCN1200
      IF(OMEGA.GE.WRGE(LK))GOTO 290                                   PTCN1201
      LST=LK                                                          PTCN1202
      IF(OMEGA.GE.WRGE(LK+1))GOTO 250                                 PTCN1203
      LST=LK+2                                                        PTCN1204
      IF(OMEGA.GE.WRGE(LK+2))GOTO 250                                 PTCN1205
      THETA=8.0                                                       PTCN1206
      GOTO 290                                                        PTCN1207
  240 THETA=0.125                                                     PTCN1208
      IF(KN.GE.7) GOTO 290                                            PTCN1209
      LK=4*KN-14                                                      PTCN1210
      IF(OMEGA.LE.WRGE(LK))GOTO 290                                   PTCN1211
      LST=2*KN-1                                                      PTCN1212
  250 THETA=ACOF(LST)+ACOF(LST+1)*ALOG(OMEGA)                         PTCN1213
      GOTO 290                                                        PTCN1214
C                                                                     PTCN1215
C  MODIFIED NEWTON METHOD FOR CORRECTOR STEPS                         PTCN1216
C                                                                     PTCN1217
  260 IF(KN.LE.1)THETA=8.0                                            PTCN1218
      IF(KN.EQ.10)THETA=1.0                                           PTCN1219
      IF(THETA.NE.0.0)GO TO 290                                       PTCN1220
      EXPON=FLOAT(KN-1)/FLOAT(KN-10)                                  PTCN1221
C                                                                     PTCN1222
C  AVOID OVERFLOW OR UNDERFLOW BY ANTICIPATING                        PTCN1223
C  CUTOFF VALUES OF THETA                                             PTCN1224
C                                                                     PTCN1225
      IF (KN.GT.10) GO TO 270                                         PTCN1226
      IF (8.0**EXPON.GT.OMEGA) THETA=8.0                              PTCN1227
      IF (.125**EXPON.LT.OMEGA) THETA=.125                            PTCN1228
      IF (THETA.NE.0.0) GO TO 290                                     PTCN1229
      GO TO 280                                                       PTCN1230
  270 IF (8.0**EXPON.LT.OMEGA) THETA=8.0                              PTCN1231
      IF (.125**EXPON.GT.OMEGA) THETA=.125                            PTCN1232
      IF (THETA.NE.0.0) GO TO 290                                     PTCN1233
  280 EXPON=1.0/EXPON                                                 PTCN1234
      THETA=OMEGA**EXPON                                              PTCN1235
      THETA=AMAX1(THETA,0.125)                                        PTCN1236
      THETA=AMIN1(THETA,8.0)                                          PTCN1237
C                                                                     PTCN1238
C  SET THE MODIFIED VALUE OF CORDIS                                   PTCN1239
C                                                                     PTCN1240
  290 CORDIS=THETA*CORDIS                                             PTCN1241
      IF(IWRITE.GE.2)WRITE(6,600)OMEGA,THETA,CORDIS                   PTCN1242
  300 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXC),1)                      PTCN1243
      CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXF),1)                      PTCN1244
      GO TO 340                                                       PTCN1245
C                                                                     PTCN1246
C*********************************************************************PTCN1247
C                                                                     PTCN1248
C  RETURNS.  SET VALUE OF IRET.  IF AN ERROR OCCURRED, PRINT          PTCN1249
C  A MESSAGE AS WELL.                                                 PTCN1250
C                                                                     PTCN1251
C*********************************************************************PTCN1252
C                                                                     PTCN1253
C                                                                     PTCN1254
C  RETURN LIMIT POINT                                                 PTCN1255
C                                                                     PTCN1256
  310 IRET=2                                                          PTCN1257
      RETURN                                                          PTCN1258
C                                                                     PTCN1259
C  RETURN WITH TARGET POINT                                           PTCN1260
```

```
C                                                                       PTCN1261
  320 IRET=1                                                            PTCN1262
      RETURN                                                            PTCN1263
C                                                                       PTCN1264
C  RETURN WITH CONTINUATION POINT                                       PTCN1265
C                                                                       PTCN1266
  330 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)                        PTCN1267
  340 IRET=0                                                            PTCN1268
      H=HTANCF                                                          PTCN1269
      RETURN                                                            PTCN1270
C                                                                       PTCN1271
C  ERROR RETURNS                                                        PTCN1272
C                                                                       PTCN1273
  350 IRET=-1                                                           PTCN1274
      IF(IWRITE.GE.1)WRITE(6,450)                                       PTCN1275
      RETURN                                                            PTCN1276
  360 IRET=-2                                                           PTCN1277
      IF(IWRITE.GE.1)WRITE(6,460)                                       PTCN1278
      RETURN                                                            PTCN1279
  370 IRET=-3                                                           PTCN1280
      IF(IWRITE.GE.1)WRITE(6,470)                                       PTCN1281
      RETURN                                                            PTCN1282
  380 IRET=-4                                                           PTCN1283
      IF(IWRITE.GE.1)WRITE(6,480)                                       PTCN1284
      RETURN                                                            PTCN1285
  390 IRET=-5                                                           PTCN1286
      IF(IWRITE.GE.1)WRITE(6,490)HTANCF,HMIN                            PTCN1287
      RETURN                                                            PTCN1288
  400 IRET=-6                                                           PTCN1289
      IF(IWRITE.GE.1)WRITE(6,500)                                       PTCN1290
      RETURN                                                            PTCN1291
  410 IRET=-7                                                           PTCN1292
      IF(IWRITE.GE.1)WRITE(6,510)                                       PTCN1293
      RETURN                                                            PTCN1294
  420 IRET=-8                                                           PTCN1295
      IF(IWRITE.GE.1)WRITE(6,520)                                       PTCN1296
      RETURN                                                            PTCN1297
  430 IRET=-9                                                           PTCN1298
      IF(IWRITE.GE.1)WRITE(6,530)                                       PTCN1299
      RETURN                                                            PTCN1300
  440 IRET=-10                                                          PTCN1301
      IF(IWRITE.GE.1)WRITE(6,540)NVAR,ISIZE                             PTCN1302
      RETURN                                                            PTCN1303
  450 FORMAT(26HOLIMIT POINT FINDER FAILED)                             PTCN1304
  460 FORMAT(50HOCORECT, SEEKING TARGET POINT, TOOK TOO MANY STEPS)     PTCN1305
  470 FORMAT(50HOCORECT, SEEKING TARGET, CALLED SOLVE WHICH FAILED)     PTCN1306
  480 FORMAT(45HOCORECT, SEEKING TARGET, FAILED WITH RAD STEP)          PTCN1307
  490 FORMAT(10HOSTEPSIZE ,F12.7,15H LESS THAN HMIN,F12.7)              PTCN1308
  500 FORMAT(42HONORM OF F(X) IS TOO LARGE ON INITIAL CALL)             PTCN1309
  510 FORMAT(33HOSOLVE FAILED IN CALL FROM TANGNT)                      PTCN1310
  520 FORMAT(33HOSOLVE FAILED IN CALL FROM CORECT)                      PTCN1311
  530 FORMAT(23HOTANGENT VECTOR IS ZERO)                                PTCN1312
  540 FORMAT(26HOUNACCEPTABLE INPUT  NVAR=,I10,7H ISIZE=,I10)           PTCN1313
  550 FORMAT(36H ANGLE BETWEEN OLD AND NEW TANGENTS=,F12.5)             PTCN1314
  560 FORMAT(9H CURVCF =,E14.6,9H CURVXF =,E14.6)                       PTCN1315
  570 FORMAT(16H USING STEPSIZE=,F12.5)                                 PTCN1316
  580 FORMAT(12H PREDICTED X/1X,5F12.5)                                 PTCN1317
  590 FORMAT(1H ,I2,16H STEP REDUCTIONS)                                PTCN1318
  600 FORMAT(7H OMEGA=,F12.5,7H THETA=,F12.5,9H NEW RAD=,F12.5)         PTCN1319
  610 FORMAT(31H TANGNT ANTICIPATES LIMIT POINT)                        PTCN1320
C                                                                       PTCN1321
C***********************************************************************PTCN1322
C                                                                       PTCN1323
      END                                                               PTCN1324
```

```
      SUBROUTINE CORECT(NVAR,X,IHOLD,WORK,IERR,IMOD,FPRYM,IPVT,          CRCT0001
     1 ABSERR,RELERR,XSTEP,NEQN,FNRM)                                    CRCT0002
C                                                                        CRCT0003
C*************************************************************************CRCT0004
C                                                                        CRCT0005
C  SUBROUTINE CORECT PERFORMS THE CORRECTOR ITERATIONS ON A STARTING     CRCT0006
C  POINT.  THE CORRECTION METHOD IS EITHER FULL (IMOD=0)                 CRCT0007
C  OR MODIFIED (IMOD=1) NEWTON'S METHOD.  FOR MODIFIED NEWTON'S          CRCT0008
C  METHOD, THE JACOBIAN IS TO BE EVALUATED ONLY AT THE STARTING POINT.   CRCT0009
C  IF B IS THE VALUE OF X(IHOLD) FOR THE INPUT STARTING POINT,           CRCT0010
C  THEN THE AUGMENTING EQUATION IS X(IHOLD)=B,                           CRCT0011
C  THAT IS, THE IHOLD-TH COMPONENT OF X IS TO BE HELD FIXED.             CRCT0012
C  THE AUGMENTED SYSTEM TO BE SOLVED IS THEN DFA(X,IHOLD)*DELTA=FA(X)    CRCT0013
C                                                                        CRCT0014
C  INPUT                                                                 CRCT0015
C  X     = THE STARTING POINT FOR THE CORRECTOR ITERATION.               CRCT0016
C  IHOLD = COMPONENT OF X THAT WILL NOT BE CHANGED DURING ITERATION      CRCT0017
C  IMOD  = FLAG FOR TYPE OF NEWTON'S METHOD TO BE USED.                  CRCT0018
C          WHEN IMOD=0, JACOBIAN IS TO BE EVALUATED AT EVERY             CRCT0019
C          CORRECTOR ITERATE.  KNMAX IS SET TO 10                        CRCT0020
C          IF IMOD=1, THE JACOBIAN IS ONLY EVALUATED AT THE STARTING     CRCT0021
C          POINT, AND KNMAX IS SET TO 20.                                CRCT0022
C                                                                        CRCT0023
C  OUTPUT                                                                CRCT0024
C  X     = SOLUTION VECTOR ON A SUCCESSFUL CALL TO CORECT.               CRCT0025
C  WORK  = THE RESIDUAL F(X), AFTER A SUCCESSFUL CALL TO CORECT.         CRCT0026
C  IERR  = THE RETURN FLAG WITH THE FOLLOWING VALUES                     CRCT0027
C         -2 MAXIMUM NUMBER OF CORRECTOR ITERATIONS WERE TAKEN.          CRCT0028
C         -1 ERROR RETURN FROM SOLVE CALLED BY CORECT.                   CRCT0029
C          0 SUCCESSFUL CORRECTION.  VECTOR X RETURNED SATISIFES         CRCT0030
C            ABS(F(X)).LE.ARSERR                                         CRCT0031
C          1 CORRECTOR STEP WAS UNACCEPTABLE, CORRECTION FAILED.         CRCT0032
C  KN    = THE NUMBER OF CORRECTOR ITERATIONS TAKEN ON THIS CALL          CRCT0033
C                                                                        CRCT0034
C  THIS SUBROUTINE IS CALLED BY                                          CRCT0035
C    PITCON                                                              CRCT0036
C  AND CALLS                                                             CRCT0037
C    SOLVE                                                               CRCT0038
C    FORTRAN ABS                                                         CRCT0039
C    LINPAK ISAMAX                                                       CRCT0040
C    LINPAK SAXPY                                                        CRCT0041
C    USER FCTN                                                           CRCT0042
C                                                                        CRCT0043
C*************************************************************************CRCT0044
C                                                                        CRCT0045
      REAL X(NVAR),WORK(NVAR),FPRYM(NVAR,NVAR)                           CRCT0046
      INTEGER IPVT(NVAR)                                                 CRCT0047
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT          CRCT0048
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM          CRCT0049
      COMMON /OUTPUT/ IWRITE                                             CRCT0050
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT                               CRCT0051
C                                                                        CRCT0052
C  INITIALIZE                                                            CRCT0053
C                                                                        CRCT0054
      KN=0                                                               CRCT0055
      KNMAX=10                                                           CRCT0056
      IF(IMOD.EQ.1)KNMAX=20                                              CRCT0057
      IERR=0                                                             CRCT0058
      FMP=2.0                                                            CRCT0059
      ICALL=1                                                            CRCT0060
      XSTEP=0.0                                                          CRCT0061
      CALL FCTN(NVAR,X,WORK)                                             CRCT0062
      IFEVAL=IFEVAL+1                                                    CRCT0063
      IMAX=ISAMAX(NEQN,WORK,1)                                           CRCT0064
      FNRM=ABS(WORK(IMAX))                                               CRCT0065
      WORK(NVAR)=0.0                                                     CRCT0066
C                                                                        CRCT0067
C  STRICTER ABSERR TEST ON STARTING POINT                                CRCT0068
C                                                                        CRCT0069
      IF (FNRM.LE.0.5*ABSERR) GO TO 60                                   CRCT0070
```

```
C                                                                       CRCT0071
C  ITERATION LOOP                                                       CRCT0072
C                                                                       CRCT0073
      DO 20 I=1, KNMAX                                                  CRCT0074
         KN=I                                                           CRCT0075
         CALL SOLVE(NVAR,X,WORK,IHOLD,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,CRCT0076
     1   IPVT)                                                          CRCT0077
         ICRSL=ICRSL+1                                                  CRCT0078
         IF(IMOD.EQ.1)ICALL=0                                           CRCT0079
         ISOLVE=ISOLVE+1                                                CRCT0080
         IF (IERR.NE.0) GO TO 50                                        CRCT0081
         FNRML=FNRM                                                     CRCT0082
         XSTEPL=XSTEP                                                   CRCT0083
         CALL SAXPY(NVAR,-1.0,WORK,1,X,1)                               CRCT0084
         IMAX=ISAMAX(NVAR,WORK,1)                                       CRCT0085
         XSTEP=ABS(WORK(IMAX))                                          CRCT0086
         IMAX=ISAMAX(NVAR,X,1)                                          CRCT0087
         XNRM=ABS(X(IMAX))                                              CRCT0088
         CALL FCTN(NVAR,X,WORK)                                         CRCT0089
         IFEVAL=IFEVAL+1                                                CRCT0090
         IMAX=ISAMAX(NEQN,WORK,1)                                       CRCT0091
         FNRM=ABS(WORK(IMAX))                                           CRCT0092
         WORK(NVAR)=0.0                                                 CRCT0093
C                                                                       CRCT0094
C  ACCEPTANCE TEST                                                      CRCT0095
C                                                                       CRCT0096
         IF (FNRM.LE.EPSATE) GO TO 60                                   CRCT0097
         IF (FNRM.GT.ABSERR) GO TO 10                                   CRCT0098
         IF (XSTEP.LE.(ABSERR+RELERR*XNRM)) GO TO 60                    CRCT0099
C                                                                       CRCT0100
C  REJECTION TEST                                                       CRCT0101
C                                                                       CRCT0102
10       IF (KN.GT.1.AND.XSTEP.GT.(FMP*XSTEPL)) GO TO 30                CRCT0103
         IF (FNRM.GT.(FMP*FNRML)) GO TO 30                              CRCT0104
20       FMP=1.05                                                       CRCT0105
      GO TO 40                                                          CRCT0106
C                                                                       CRCT0107
C  UNSUCCESSFUL STEP                                                    CRCT0108
C                                                                       CRCT0109
30    IERR=-3                                                           CRCT0110
      IF(IWRITE.EQ.2)WRITE(6,120)                                       CRCT0111
      GO TO 70                                                          CRCT0112
C                                                                       CRCT0113
C  MAXIMUM NUMBER OF CORRECTOR STEPS REACHED                            CRCT0114
C                                                                       CRCT0115
40    IERR=-2                                                           CRCT0116
      IF(IWRITE.EQ.2)WRITE(6,110)                                       CRCT0117
      GO TO 70                                                          CRCT0118
C                                                                       CRCT0119
C  ERROR RETURN IN SOLVE                                                CRCT0120
C                                                                       CRCT0121
50    IERR=-1                                                           CRCT0122
      IF(IWRITE.EQ.2)WRITE(6,100)                                       CRCT0123
      GO TO 70                                                          CRCT0124
C                                                                       CRCT0125
C  SUCCESSFUL STEP                                                      CRCT0126
C                                                                       CRCT0127
60    IERR=0                                                            CRCT0128
70    KNSUM=KNSUM+KN                                                    CRCT0129
      IF(IWRITE.EQ.2)WRITE(6,80) KN,XSTEP                               CRCT0130
      IF(IWRITE.EQ.2)WRITE(6,90)IHOLD                                   CRCT0131
      RETURN                                                            CRCT0132
80    FORMAT(13H CORECT TOOK ,I2,21H STEPS, LAST ONE WAS ,E12.5)        CRCT0133
90    FORMAT(14H CORECT IHOLD=,I3)                                      CRCT0134
100   FORMAT(31HOSOLVE FAILED, CALLED BY CORECT)                        CRCT0135
110   FORMAT(25HOTOO MANY CORRECTOR STEPS)                              CRCT0136
120   FORMAT(24HOCORRECTOR STEP REJECTED)                               CRCT0137
C                                                                       CRCT0138
C***********************************************************************CRCT0139
C                                                                       CRCT0140
      END                                                               CRCT0141
```

```
      SUBROUTINE TANGNT(NVAR,X,IP,TAN,IRET,ICALL,FPRYM,IPVT,NEQN,DETA,   TNGN0001
     1 IEXP)                                                             TNGN0002
C                                                                        TNGN0003
C************************************************************************TNGN0004
C                                                                        TNGN0005
C  SUBROUTINE TANGNT COMPUTES THE UNIT TANGENT VECTOR TO THE SOLUTION    TNGN0006
C  CURVE OF THE UNDERDETERMINED NONLINEAR SYSTEM FX = 0.  THE            TNGN0007
C  TANGENT VECTOR TAN IS THE SOLUTION OF THE LINEAR SYSTEM               TNGN0008
C                                                                        TNGN0009
C        DFA(X,IPL)*TAN = E(NVAR)                                        TNGN0010
C                                                                        TNGN0011
C  WHERE DFA(X,IPL) IS THE NVAR BY NVAR MATRIX WHOSE FIRST  NVAR-1  ROWSTNGN0012
C  ARE DFX/DX (X), THE DERIVATIVE OF FX EVALUATED AT X, AND WHOSE LAST   TNGN0013
C  ROW IS (E(IPL)) TRANSPOSE, THE NVAR COMPONENT EUCLIDEAN COORDINATE    TNGN0014
C  VECTOR WITH 1 IN THE IPL-TH POSITION AND ZEROS ELSEWHERE.  E(NVAR) ISTNGN0015
C  THE NVAR COMPONENT EUCLIDEAN COORDINATE VECTOR WITH ONE IN THE        TNGN0016
C  LAST COMPONENT.                                                       TNGN0017
C    THE TANGENT VECTOR IS THEN NORMALIZED AND ITS SIGN ADJUSTED.        TNGN0018
C                                                                        TNGN0019
C  INPUT                                                                 TNGN0020
C  NVAR   = THE NUMBER OF VARIABLES                                      TNGN0021
C  X      = THE CURRENT CONTINUATION POINT                               TNGN0022
C  IP     = CONTINUATION COMPONENT SET ON LAST STEP                      TNGN0023
C                                                                        TNGN0024
C  OUTPUT                                                                TNGN0025
C  TAN    = THE UNIT TANGENT VECTOR IN CONTINUATION DIRECTION AT X       TNGN0026
C  DETA   = BINARY MANTISSA OF DETERMINANT OF JACOBIAN DFA(X,IPL)        TNGN0027
C  IEXP   = BINARY EXPONENT OF THE DETERMINANT OF JACOBIAN DFA(X,IPL)    TNGN0028
C  IP     = LOCATION OF LARGEST COMPONENT OF TANGENT VECTOR TAN          TNGN0029
C           CANDIDATE FOR NEW CONTINUATION COMPONENT                     TNGN0030
C                                                                        TNGN0031
C  THIS SUBROUTINE IS CALLED BY                                          TNGN0032
C    PITCON                                                              TNGN0033
C  AND CALLS                                                             TNGN0034
C    SOLVE                                                               TNGN0035
C    LINPAK ISAMAX                                                       TNGN0036
C    LINPAK SNRM2                                                        TNGN0037
C    LINPAK SSCAL                                                        TNGN0038
C                                                                        TNGN0039
C************************************************************************TNGN0040
C                                                                        TNGN0041
      REAL X(NVAR),TAN(NVAR),FPRYM(NVAR,NVAR)                            TNGN0042
      INTEGER IPVT(NVAR)                                                 TNGN0043
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT          TNGN0044
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM          TNGN0045
      COMMON /OUTPUT/ IWRITE                                             TNGN0046
C                                                                        TNGN0047
C  COMPUTE TANGENT VECTOR                                                TNGN0048
C                                                                        TNGN0049
      DO 10 I=1,NEQN                                                     TNGN0050
   10    TAN(I)=0.0                                                      TNGN0051
      TAN(NVAR)=1.0                                                      TNGN0052
      IERR=0                                                             TNGN0053
      CALL SOLVE(NVAR,X,TAN,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,          TNGN0054
     1 IPVT)                                                             TNGN0055
      ITNSL=ITNSL+1                                                      TNGN0056
      ISOLVE=ISOLVE+1                                                    TNGN0057
      IF (IERR.NE.0) IRET=-1                                             TNGN0058
      IF (IRET.LT.0) RETURN                                              TNGN0059
C                                                                        TNGN0060
C  OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR                               TNGN0061
C                                                                        TNGN0062
      IP=ISAMAX(NVAR,TAN,1)                                              TNGN0063
      TNORM=SNRM2(NVAR,TAN,1)                                            TNGN0064
      IF (TNORM.EQ.0.0) IRET=-2                                          TNGN0065
      IF (IRET.LT.0) RETURN                                              TNGN0066
C                                                                        TNGN0067
C  NORMALIZE THE VECTOR                                                  TNGN0068
C                                                                        TNGN0069
      SCALER=1.0/TNORM                                                   TNGN0070
      CALL SSCAL(NVAR,SCALER,TAN,1)                                      TNGN0071
      RETURN                                                             TNGN0072
C                                                                        TNGN0073
C************************************************************************TNGN0074
C                                                                        TNGN0075
      END                                                                TNGN0076
```

```
      SUBROUTINE ROOT(A,FA,B,FB,C,FC,KOUNT,IFLAG)                       ROOT0001
C                                                                       ROOT0002
C***********************************************************************ROOT0003
C                                                                       ROOT0004
C  SUBROUTINE ROOT SEEKS A ROOT OF THE EQUATION F(X)=0.0,               ROOT0005
C  GIVEN A STARTING INTERVAL (A,C) ON WHICH F CHANGES SIGN.             ROOT0006
C  ON FIRST CALL TO ROOT, THE INTERVAL AND FUNCTION VALUES              ROOT0007
C  FA AND FC ARE FED IN AND AN APPROXIMATION B FOR THE ROOT IS RETURNED.ROOT0008
C  BEFORE EACH SUBSEQUENT CALL, THE USER EVALUATES FB=F(B), AND THE     ROOT0009
C  PROGRAM TRIES TO RETURN A BETTER APPROXIMATION B.                    ROOT0010
C                                                                       ROOT0011
C  THIS PROGRAM IS BASED ON THE FORTRAN FUNCTION ZERO                   ROOT0012
C  GIVEN IN THE BOOK:                                                   ROOT0013
C  'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES'                    ROOT0014
C  BY RICHARD P. BRENT, PRENTICE HALL,INC, 1973                         ROOT0015
C                                                                       ROOT0016
C  THE MODIFICATIONS WERE DONE BY JOHN BURKARDT.                        ROOT0017
C                                                                       ROOT0018
C  ON INPUT:                                                            ROOT0019
C                                                                       ROOT0020
C  A       - IS ONE ENDPOINT OF AN INTERVAL IN WHICH F CHANGES SIGN.    ROOT0021
C  FA      - THE VALUE OF F(A).  THE USER MUST EVALUATE F(A) BEFORE FIRSTROOT0022
C            CALL ONLY.  THEREAFTER THE PROGRAM SETS FA.                ROOT0023
C  B       - ON FIRST CALL, B SHOULD NOT BE SET BY THE USER.            ROOT0024
C            ON SUBSEQUENT CALLS, B SHOULD NOT BE CHANGED               ROOT0025
C            FROM ITS OUTPUT VALUE, THE CURRENT APPROXIMANT             ROOT0026
C            TO THE ROOT.                                               ROOT0027
C  FB      - ON FIRST CALL, FB SHOULD NOT BE SET BY THE USER.           ROOT0028
C            THEREAFTER, THE USER SHOULD EVALUATE THE FUNCTION          ROOT0029
C            AT THE OUTPUT VALUE B, AND RETURN FB=F(B).                 ROOT0030
C  C       - IS THE OTHER ENDPOINT OF THE INTERVAL IN WHICH             ROOT0031
C            F CHANGES SIGN.  NOTE THAT THE PROGRAM WILL RETURN         ROOT0032
C            IMMEDIATELY WITH AN ERROR FLAG IF FC*FA.GT.0.0.            ROOT0033
C  FC      - THE VALUE OF F(C).  THE USER MUST EVALUATE F(C) BEFORE FIRSTROOT0034
C            CALL ONLY.  THEREAFTER THE PROGRAM SETS FC.                ROOT0035
C  KOUNT   - A COUNTER FOR THE NUMBER OF CALLS TO ROOT.  KOUNT          ROOT0036
C            SHOULD BE SET TO ZERO ON THE FIRST CALL FOR A GIVEN        ROOT0037
C            ROOT PROBLEM.                                              ROOT0038
C  IFLAG   - AN ERROR RETURN FLAG WHOSE INPUT VALUE IS IMMATERIAL.      ROOT0039
C                                                                       ROOT0040
C  ON RETURN FROM A CALL TO ROOT                                        ROOT0041
C                                                                       ROOT0042
C  A       - ONE ENDPOINT OF CHANGE OF SIGN INTERVAL.                   ROOT0043
C  FA      - THE VALUE OF F(A).                                         ROOT0044
C  B       - CURRENT APPROXIMATION TO THE ROOT.  BEFORE ANOTHER         ROOT0045
C            CALL TO ROOT, EVALUATE F(B).                               ROOT0046
C  FB      - FB WILL BE OVERWRITTEN BY THE USER BEFORE ANOTHER          ROOT0047
C            CALL.  ITS VALUE ON RETURN IS ONE OF FA, FB OR FC.         ROOT0048
C  C       - OTHER ENDPOINT OF CHANGE IN SIGN INTERVAL.                 ROOT0049
C  FC      - THE VALUE OF F(C).                                         ROOT0050
C  KOUNT   - CURRENT NUMBER OF CALLS TO ROOT.                           ROOT0051
C  IFLAG   - PROGRAM RETURN FLAG:                                       ROOT0052
C            IFLAG=-2 MEANS THAT ON FIRST CALL, FA*FC.GT.0.0.           ROOT0053
C                     THIS IS AN ERROR RETURN, SINCE A BRACKETING       ROOT0054
C                     INTERVAL SHOULD BE SUPPLIED ON FIRST CALL.        ROOT0055
C            IFLAG=-1 MEANS THAT THE CURRENT BRACKETING INTERVAL        ROOT0056
C                     WHOSE ENDPOINTS ARE STORED IN A AND C             ROOT0057
C                     IS SO SMALL (LESS THAN 4*EPMACH*ABS(B)+EPMACH)    ROOT0058
C                     THAT B SHOULD BE ACCEPTED AS THE ROOT.            ROOT0059
C                     THE FUNCTION VALUE F(B) IS STORED IN FB.          ROOT0060
C            IFLAG= 0 MEANS THAT THE INPUT VALUE FB IS EXACTLY          ROOT0061
C                     ZERO, AND B SHOULD BE ACCEPTED AS THE ROOT.       ROOT0062
C                                                                       ROOT0063
C            IFLAG.GT.0  MEANS THAT THE CURRENT APPROXIMATION TO        ROOT0064
C                     THE ROOT IS CONTAINED IN B.  IF A BETTER          ROOT0065
C                     APPROXIMATION IS DESIRED, SET FB=F(B)             ROOT0066
C                     AND CALL ROOT AGAIN.  THE VALUE OF IFLAG INDICATES ROOT0067
C                     THE METHOD THAT WAS USED TO PRODUCE B.            ROOT0068
C                                                                       ROOT0069
C            IFLAG= 1 BISECTION WAS USED.                               ROOT0070
```

```
C           IFLAG= 2 LINEAR INTERPOLATION (SECANT METHOD).                  ROOT0071
C           IFLAG= 3 INVERSE QUADRATIC INTERPOLATION.                       ROOT0072
C                                                                           ROOT0073
C  LOCAL VARIABLES INCLUDE:                                                 ROOT0074
C                                                                           ROOT0075
C  EPMACH- SMALLEST POSITIVE NUMBER SUCH THAT 1.0+EPMACH.GT.1.0             ROOT0076
C          .5*BETA**(1-TAU) FOR ROUNDED, TAU-DIGIT ARITHMETIC               ROOT0077
C          BASE BETA.  TWICE THAT VALUE FOR TRUNCATED ARITHMETIC.           ROOT0078
C          THIS IS THE RELATIVE MACHINE PRECISION.                          ROOT0079
C  HALFBC- SIGNED HALFWIDTH OF INTERVAL.  DURING SEGMENT 3, THE             ROOT0080
C          CHANGE OF SIGN INTERVAL IS (B,C) OR (C,B).  THE MIDPOINT         ROOT0081
C          OF THAT INTERVAL IS XMID=B+HALFBC, REGARDLESS OF ORIENTATION.    ROOT0082
C  SDEL1 - SIZE OF CHANGE IN SIGN INTERVAL.                                 ROOT0083
C  SDEL2 - PREVIOUS VALUE OF SDEL1.                                         ROOT0084
C  SDEL3 - PREVIOUS VALUE OF SDEL2.                                         ROOT0085
C  SDEL4 - PREVIOUS VALUE OF SDEL3.                                         ROOT0086
C  STEP  - THE NEW ROOT IS COMPUTED AS A CORRECTION TO B OF THE             ROOT0087
C          FORM B(NEW)=B(OLD)+STEP.                                         ROOT0088
C  TOLER - A NUMBER WE ACCEPT AS 'SMALL' WHEN EXAMINING INTERVAL            ROOT0089
C          SIZE OR STEP SIZE.  TOLER=2.0*EPMACH*ABS(B) + EPMACH IS          ROOT0090
C          A MINIMUM BELOW WHICH WE WILL NOT ALLOW SUCH VALUES TO FALL.     ROOT0091
C  THIS SUBROUTINE IS CALLED BY                                             ROOT0092
C    PITCON                                                                 ROOT0093
C  AND CALLS                                                                ROOT0094
C    FORTRAN ABS                                                            ROOT0095
C    FORTRAN SIGN                                                           ROOT0096
C                                                                           ROOT0097
C***************************************************************************ROOT0098
C                                                                           ROOT0099
      REAL A,B,C,FA,FB,FC,STEP,TOLER,P,Q,R,S                                ROOT0100
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT                                  ROOT0101
C                                                                           ROOT0102
C  SEGMENT 1:  FIRST CALL HANDLED SPECIALLY.  DO BOOKKEEPING.               ROOT0103
C                                                                           ROOT0104
C  SET CERTAIN VALUES ONLY FOR INITIAL CALL WITH KOUNT=0                    ROOT0105
C                                                                           ROOT0106
      IF (KOUNT.GT.0) GO TO 10                                              ROOT0107
      IF (FA.GT.0.0.AND.FC.GT.0.0) GO TO 110                                ROOT0108
      IF (FA.LT.0.0.AND.FC.LT.0.0) GO TO 110                                ROOT0109
      KOUNT=1                                                               ROOT0110
      SDEL1=2.0*ABS(C-A)                                                    ROOT0111
      SDEL2=2.0*SDEL1                                                       ROOT0112
      SDEL3=2.0*SDEL2                                                       ROOT0113
      B=C                                                                   ROOT0114
      FB=FC                                                                 ROOT0115
      GO TO 20                                                              ROOT0116
C                                                                           ROOT0117
C  ON EVERY CALL, INCREMENT COUNTER                                         ROOT0118
C                                                                           ROOT0119
   10 KOUNT=KOUNT+1                                                         ROOT0120
C                                                                           ROOT0121
C  RETURN IF HIT MACHINE ZERO FOR F(B)                                      ROOT0122
C                                                                           ROOT0123
      IF(FB.EQ.0.0) GO TO 90                                                ROOT0124
C                                                                           ROOT0125
C  SEGMENT 2:  REARRANGE POINTS AND FUNCTION VALUES IF                      ROOT0126
C  NECESSARY SO THAT FB*FC.LT.0.0, AND SO THAT                              ROOT0127
C  ABS(FB).LT.ABS(FC)                                                       ROOT0128
C                                                                           ROOT0129
      IF((FB.LE.0.0).AND.(FC.GT.0.0)) GO TO 30                              ROOT0130
      IF((FB.GT.0.0).AND.(FC.LE.0.0)) GO TO 30                              ROOT0131
C                                                                           ROOT0132
C  FB AND FC ARE OF SAME SIGN.                                              ROOT0133
C  (ROOT CHANGED SIGN)                                                      ROOT0134
C  OVERWRITE C WITH VALUE OF A                                              ROOT0135
C                                                                           ROOT0136
   20 C=A                                                                   ROOT0137
      FC=FA                                                                 ROOT0138
```

```
C                                                                       ROOT0139
C  IF NECESSARY, SET A:=B, B:=C, C:=B                                   ROOT0140
C  TO ENSURE THAT ABS(FB).LE.ABS(FC)                                    ROOT0141
C                                                                       ROOT0142
   30 IF(ABS(FC).GE.ABS(FB)) GO TO 40                                   ROOT0143
      A=B                                                               ROOT0144
      B=C                                                               ROOT0145
      C=A                                                               ROOT0146
      FA=FB                                                             ROOT0147
      FB=FC                                                             ROOT0148
      FC=FA                                                             ROOT0149
C                                                                       ROOT0150
C  SEGMENT 3:  CHECK FOR ACCEPTANCE BECAUSE OF SMALL INTERVAL.          ROOT0151
C  CURRENT CHANGE IN SIGN INTERVAL IS (C,B) OR (B,C).                   ROOT0152
C                                                                       ROOT0153
   40 TOLER=2.0*EPMACH*ABS(B)+EPMACH                                    ROOT0154
      HALFBC=0.5*(C-B)                                                  ROOT0155
      SDEL4=SDEL3                                                       ROOT0156
      SDEL3=SDEL2                                                       ROOT0157
      SDEL2=SDEL1                                                       ROOT0158
      SDEL1=ABS(C-B)                                                    ROOT0159
      IF(ABS(HALFBC).LE.TOLER) GO TO 100                                ROOT0160
C                                                                       ROOT0161
C  SEGMENT 4:  COMPUTE NEW APPROXIMANT TO ROOT OF THE FORM              ROOT0162
C  B(NEW)=B(OLD)+STEP.                                                  ROOT0163
C  METHODS AVAILABLE ARE LINEAR INTERPOLATION                           ROOT0164
C  INVERSE QUADRATIC INTERPOLATION                                      ROOT0165
C  AND BISECTION.                                                       ROOT0166
C                                                                       ROOT0167
      IF(ABS(FB).GE.ABS(FA))GO TO 70                                    ROOT0168
      IF(A.NE.C) GO TO 50                                               ROOT0169
C                                                                       ROOT0170
C  ATTEMPT LINEAR INTERPOLATION IF ONLY TWO POINTS AVAILABLE            ROOT0171
C  COMPUTE P AND Q FOR APPROXIMATION B(NEW)=B(OLD)+P/Q                  ROOT0172
C                                                                       ROOT0173
      IFLAG=2                                                           ROOT0174
      S=FB/FA                                                           ROOT0175
      P=2.0*HALFBC*S                                                    ROOT0176
      Q=1.0-S                                                           ROOT0177
      GO TO 60                                                          ROOT0178
C                                                                       ROOT0179
C  ATTEMPT INVERSE QUADRATIC INTERPOLATION IF THREE POINTS AVAILABLE    ROOT0180
C  COMPUTE P AND Q FOR APPROXIMATION B(NEW)=B(OLD)+P/Q                  ROOT0181
C                                                                       ROOT0182
   50 IFLAG=3                                                           ROOT0183
      S=FB/FA                                                           ROOT0184
      Q=FA/FC                                                           ROOT0185
      R=FB/FC                                                           ROOT0186
      P=S*(2.0*HALFBC*Q*(Q-R)-(B-A)*(R-1.0))                            ROOT0187
      Q=(Q-1.0)*(R-1.0)*(S-1.0)                                         ROOT0188
C                                                                       ROOT0189
C  CORRECT THE SIGNS OF P AND Q                                         ROOT0190
C                                                                       ROOT0191
   60 IF(P.GT.0.0)Q=-Q                                                  ROOT0192
      P=ABS(P)                                                          ROOT0193
C                                                                       ROOT0194
C  IF P/Q IS TOO LARGE, GO BACK TO BISECTION                            ROOT0195
C                                                                       ROOT0196
      IF(8.0*SDEL1.GT.SDEL4) GO TO 70                                   ROOT0197
      IF (P.GE.1.5*ABS(HALFBC*Q)-ABS(TOLER*Q)) GO TO 70                 ROOT0198
      STEP=P/Q                                                          ROOT0199
      GO TO 80                                                          ROOT0200
C                                                                       ROOT0201
C  PERFORM BISECTION:                                                   ROOT0202
C   IF ABS(FB).GE.ABS(FA)                                               ROOT0203
C   OR INTERPOLATION IS UNSAFE (P/Q IS LARGE)                           ROOT0204
C   OR IF THREE CONSECUTIVE STEPS HAVE NOT DECREASED                    ROOT0205
C      THE SIZE OF THE INTERVAL BY A FACTOR OF 8.0                      ROOT0206
C                                                                       ROOT0207
   70 IFLAG=1                                                           ROOT0208
      STEP=HALFBC                                                       ROOT0209
      GO TO 80                                                          ROOT0210
```

```
C                                                                       ROOT0211
C  SEGMENT 5:  VALUE OF STEP HAS BEEN COMPUTED.                         ROOT0212
C  UPDATE INFORMATION: A:=B, FA:=FB, B:=B+STEP.                         ROOT0213
C  CHANGE IN SIGN INTERVAL IS NOW (A,C) OR (C,A).                       ROOT0214
C                                                                       ROOT0215
   80 A=B                                                               ROOT0216
      FA=FB                                                             ROOT0217
      IF(ABS(STEP).LE.TOLER) STEP=SIGN(TOLER,HALFBC)                    ROOT0218
      B=B+STEP                                                          ROOT0219
      RETURN                                                            ROOT0220
C                                                                       ROOT0221
C  SPECIAL RETURNS                                                      ROOT0222
C                                                                       ROOT0223
C  INPUT POINT B IS EXACT ROOT                                          ROOT0224
C                                                                       ROOT0225
   90 IFLAG=0                                                           ROOT0226
      RETURN                                                            ROOT0227
C                                                                       ROOT0228
C  CHANGE IN SIGN INTERVAL IS OF SIZE LESS THAN 4*EPMACH*ABS(B)+EPMACH  ROOT0229
C  INTERVAL RETURNED AS (B,C) OR (C,B).                                 ROOT0230
C  ACCEPT B AS ROOT WITH RESIDUAL F(B) STORED IN FB.                    ROOT0231
C                                                                       ROOT0232
  100 IFLAG=-1                                                          ROOT0233
      A=B                                                               ROOT0234
      FA=FB                                                             ROOT0235
      RETURN                                                            ROOT0236
C                                                                       ROOT0237
C  CHANGE OF SIGN CONDITION VIOLATED                                    ROOT0238
C                                                                       ROOT0239
  110 IFLAG=-2                                                          ROOT0240
      KOUNT=0                                                           ROOT0241
      RETURN                                                            ROOT0242
C                                                                       ROOT0243
C*************************************************************************ROOT0244
C                                                                       ROOT0245
      END                                                               ROOT0246
```

```
      SUBROUTINE SOLVE(NVAR,X,Y,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,     SLVE0001
     1 IPVT)                                                            SLVE0002
C                                                                       SLVE0003
C***********************************************************************SLVE0004
C                                                                       SLVE0005
C  THIS SUBROUTINE IS CALLED BY                                         SLVE0006
C    CORECT                                                             SLVE0007
C    TANGNT                                                             SLVE0008
C  AND CALLS                                                            SLVE0009
C    FORTRAN ABS                                                        SLVE0010
C    LINPAK SGEFA                                                       SLVE0011
C    LINPAK SGESL                                                       SLVE0012
C    USER FPRIME                                                        SLVE0013
C                                                                       SLVE0014
C  THIS SUBROUTINE SOLVES THE LINEAR SYSTEM  DFA(X,IP)*YOUT = YIN       SLVE0015
C  WHERE DFA(X,IP) IS THE (NVAR)X(NVAR) MATRIX WHOSE FIRST NVAR - 1     SLVE0016
C  ROWS ARE THE JACOBIAN COMPUTED BY FPRIME, AND WHOSE LAST             SLVE0017
C  ROW IS ALL 0 EXCEPT FOR A 1 IN THE IP-TH COMPONENT.                  SLVE0018
C                                                                       SLVE0019
C  YIN IS THE NVAR COMPONENT VECTOR Y ON INPUT, AND THE SOLUTION        SLVE0020
C  VECTOR YOUT IS RETURNED IN Y ON OUTPUT AFTER A SUCCESSFUL            SLVE0021
C  SETUP AND SOLUTION.                                                  SLVE0022
C                                                                       SLVE0023
C  **NOTE** SUBROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE      SLVE0024
C  LINEAR SYSTEM.  THE USER MAY WISH TO REPLACE THIS ROUTINE WITH       SLVE0025
C  ONE MORE SUITED TO HIS PRORLEM.                                      SLVE0026
C                                                                       SLVE0027
C  DETA    BINARY MANTISSA OF THE DETERMINANT OF JACOBIAN DFA(X,IP)     SLVE0028
C  IEXP    BINARY EXPONENT OF THE DETERMINANT OF JACOBIAN DFA(X,IP)     SLVE0029
C  IMOD    NEWTON METHOD FLAG.                                          SLVE0030
C          IMOD=0, JACOBIAN IS TO BE EVALUATED FOR EVERY CORRECTOR STEP SLVE0031
C                  AND EVERY TANGENT CALCULATION                        SLVE0032
C          IMOD=1, JACOBIAN IS TO BE EVALUATED ONLY FOR FIRST CORRECTOR SLVE0033
C                  STEP, AND EVERY TANGENT CALCULATION                  SLVE0034
C  ICALL   SET UP FLAG.                                                 SLVE0035
C          IF (ICALL.EQ.0.AND.IMOD.NE.0) DON'T RE-EVALUATED JACORIAN    SLVE0036
C  INFO    OUTPUT FROM SGEFA.  IF INFO.NE.0, SGEFA FOUND A ZERO         SLVE0037
C          PIVOT WHEN ELIMINATING INFO-TH VARIABLE.                     SLVE0038
C  IERR    RETURN FLAG, 0 MEANS SUCCESSFUL SOLUTION, 1 MEANS FAILURE    SLVE0039
C  NVAR    THE NUMBER OF VARIABLES IN THE NONLINEAR SYSTEM              SLVE0040
C  X       THE POINT AT WHICH TO EVALUATE FPRYM                         SLVE0041
C  Y       THE RIGHT HAND SIDE ON INPUT, THE SOLUTION                   SLVE0042
C          ON OUTPUT                                                    SLVE0043
C  FPRYM   ARRAY WHERE DFA(X,IP) IS TO BE STORED.                       SLVE0044
C  IPVT    INTEGER WORK SPACE FOR PIVOT ROW SWITCHES DEMANDED BY SGEFA  SLVE0045
C  IP      THE VARIABLE USED IN THE AUGMENTING EQUATION THAT IS OF THE  SLVE0046
C          FORM X(IP)=B.  HENCE THE LAST ROW OF DFA(X,IP) IS ALL        SLVE0047
C          ZERO EXCEPT FOR A 1.0 IN THE IP-TH COLUMN.                   SLVE0048
C                                                                       SLVE0049
C***********************************************************************SLVE0050
C                                                                       SLVE0051
      REAL X(NVAR),Y(NVAR),FPRYM(NVAR,NVAR)                             SLVE0052
      INTEGER IPVT(NVAR)                                                SLVE0053
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM         SLVE0054
C                                                                       SLVE0055
C  DEPENDING ON VALUES OF ICALL AND IMOD, EITHER SET UP                 SLVE0056
C  AUGMENTED JACORIAN, DECOMPOSE INTO L-U FACTORS, AND GET DETERMINANT, SLVE0057
C  OR USE CURRENT FACTORED JACORIAN WITH NEW RIGHT-HAND SIDE.           SLVE0058
C                                                                       SLVE0059
      IF (ICALL.EQ.0.AND.IMOD.NE.0) GO TO 50                            SLVE0060
      CALL FPRIME(NVAR,X,FPRYM)                                         SLVE0061
      IPEVAL=IPEVAL+1                                                   SLVE0062
      DO 10 I=1,NVAR                                                    SLVE0063
   10    FPRYM(NVAR,I)=0.0                                              SLVE0064
      FPRYM(NVAR,IP)=1.0                                                SLVE0065
C                                                                       SLVE0066
C  CARRY OUT IN CORE LU DECOMPOSITION OF NVAR BY NVAR MATRIX            SLVE0067
C  AND USE PIVOT INFORMATION TO COMPUTE DETERMINANT                     SLVE0068
```

```
C                                                                       SLVE0069
      CALL SGEFA(FPRYM,NVAR,NVAR,IPVT,INFO)                             SLVE0070
      DETA=1.0                                                          SLVE0071
      IEXP=0                                                            SLVE0072
      TWO=2.0                                                           SLVE0073
      DO 40 I=1,NVAR                                                    SLVE0074
           IF (IPVT(I).NE.I) DETA=-DETA                                 SLVE0075
           DETA=FPRYM(I,I)*DETA                                         SLVE0076
           IF (DETA.EQ.0.0) GO TO 60                                    SLVE0077
   20      IF (ABS(DETA).GE.1.0) GO TO 30                               SLVE0078
           DETA=DETA*TWO                                                SLVE0079
           IEXP=IEXP-1                                                  SLVE0080
           GO TO 20                                                     SLVE0081
   30      IF (ABS(DETA).LT.TWO) GO TO 40                               SLVE0082
           DETA=DETA/TWO                                                SLVE0083
           IEXP=IEXP+1                                                  SLVE0084
           GO TO 30                                                     SLVE0085
   40      CONTINUE                                                     SLVE0086
      IF (INFO.NE.0) GO TO 60                                           SLVE0087
C                                                                       SLVE0088
C  USING L-U FACTORED MATRIX, SOLVE SYSTEM USING FORWARD-BACKWARD       SLVE0089
C  ELIMINATION, AND OVERWRITE RIGHT HAND SIDE WITH SOLUTION             SLVE0090
C                                                                       SLVE0091
   50 CALL SGESL(FPRYM,NVAR,NVAR,IPVT,Y,0)                              SLVE0092
      IERR=0                                                            SLVE0093
      RETURN                                                            SLVE0094
   60 IERR=1                                                            SLVE0095
      INFO=0                                                            SLVE0096
      RETURN                                                            SLVE0097
C                                                                       SLVE0098
C***********************************************************************SLVE0099
C                                                                       SLVE0100
      END                                                               SLVE0101
```

REFERENCES

1. J. P. Abbott, An efficient algorithm for the determination of certain bifurcation points, J. of Computational and Applied Math. 4, 1978, 19-27.

2. E. Allgower and K. Georg, Simplicial and continuation methods for approximating fixed points and solutions to systems of equations, SIAM Review 22, 1980, 28-85.

3. P. T. Boggs, The solution of nonlinear systems of equations by A-stable integration techniques, SIAM J. Num. Anal. 8, 1971, 767-785.

4. R. P. Brent, Algorithms for minimization without derivatives, Prentice Hall, Englewood Cliffs, NJ, 1973.

5. M. A. Crisfield, A fast incremental/iterative solution procedure that handles snap-through, Computers and Structures 13, 1981, 55-62.

6. D. F. Davidenko, On a new method of numerical solution of systems of nonlinear equations, Dokl. Akad. Nauk USSR 88, 1953, 601-602.

7. C. Den Heijer and W. C. Rheinboldt, On steplength algorithms for a class of continuation methods, SIAM J. Num. Anal. 18, 1981, 925-947.

8. F. Ficken, The continuation method for functional equations, Comm. Pure Appl. Math. 4, 1951, 435-456.

9. F. Freudenstein and B. Roth, Numerical solution of systems of nonlinear equations, J. ACM 10, 1963, 550-556.

10. H. B. Keller, Numerical solution of bifurcation and nonlinear eigenvalue problems, in "Applications of Bifurcation Theory", ed. by P. Rabinowitz, Academic Press, New York, NY, 1977, 359-384.

11. H. B. Keller, Global homotopies and Newton methods in "Recent Advances in Numerical Analysis", ed. by C. deBoor, G. H. Golub, Academic Press, New York, NY, 1978, 73-94.

12. A. D. Kerr and M. T. Soifer, The linearization of the prebuckling state and its effect on the determined instability load, Trans. ASME, J. of Applied Mech., V. 36, 1969, 775-783.

13. W. Kizner, A numerical method for finding solutions of nonlinear equations, SIAM J. Appl. Math. 12, 1964, 424-428.

14. M. Kubicek, Algorithm 502, Dependence of solution of nonlinear systems on a parameter, ACM-TOMS 2, 1976, 98-107.

15. M. Kubicek, M. Holodniok and I. Marek, Numerical solution of nonlinear equations by one-parameter imbedding methods, Num. Functional Anal. and Optim. 3, 1981, 223-264.

16. S. T. Mau and R. H. Gallagher, A finite element procedure for nonlinear prebuckling and initial postbuckling analysis, NASA Contractor Report, NASA-CR_1936, January 1972.

17. J. W. Milnor, Topology from the differential viewpoint, Univ. of Virginia Press, Charlottesville, VA, 1965.

18. R. K. Mehra, W. C. Kessel and J. V. Carroll, Global stability and control analysis of aircraft at high angles of attack, ONR Report-CR-215-248 1,2,3, June 1977,78,79.

19. R. Menzel and H. Schwetlick, Zur Lösung parameter-abhängiger nichtlinearer Gleichungen mit singulären Jacobi-Matrizen, Num. Math. 30, 1978, 65-79.

20. G. Moore and A. Spence, The calculation of turning points of nonlinear equations, SIAM J. Num. Anal. 17, 1980, 567-576.

21. J. T. Oden, Finite elements of nonlinear continua, McGraw Hill, New York, 1972.

22. G. Pönisch and H. Schwetlick, Computing turning points of curves implicitly defined by nonlinear equations depending on a parameter, Computing 26, 1981, 107-121.

23. T. Poston and I. Stewart, Catastrophe theory and its applications, Pitman Publ. Ltd, London, 1978.

24. W. C. Rheinboldt, An adaptive continuation process for solving systems of nonlinear equations, Polish Academy of Science, Banach Ctr. Publ., Vol. 3, 1977, 129-142.

25. W. C. Rheinboldt, Numerical methods for a class of finite dimensional bifurcation problems, SIAM J. Num. Anal. 15, 1978, 1-11.

26. W. C. Rheinboldt, Solution fields of nonlinear equations and continuation methods, SIAM J. Num. Anal. 17, 1980, 221-237.

27. W. C. Rheinboldt, Numerical analysis of continuation methods for nonlinear structural problems, Computers and Structures 13, 1981, 103-114.

28. W. C. Rheinboldt, Computation of critical boundaries on equilibrium manifolds, Univ. of Pittsburgh, Inst. f. Comp. Math. and Appl., Techn. Rept. ICMA-80-20, 1980; SIAM J. Num. Anal., 1981, in press.

29. E. Riks, An incremental approach to the solution of snapping and buckling problems, Int. J. Solids and Structures 15, 1979, 524-551.

30. A. A. Schy and M. E. Hannah, Prediction of jump phenomena in roll-coupled maneuvers of airplanes, J. of Aircraft 14, 1977, 375-382.

31. M. J. Sewell, On the connection between stability and the shape of the equilibrium surface, J. Mech. Phys. Solids 14, 1966, 203-230.

32. M. J. Sewell, Some global equilibrium surfaces, Int. J. Mech. Engng. Educ. 6, 1978, 163-174.

33. R. Seydel, Numerical computation of branch points in nonlinear equations, Numer. Math. 33, 1979, 339-352.

34. R. B. Simpson, A method for the numerical determination of bifurcation states of nonlinear systems of equations, SIAM J. Num. Anal. 12, 1975, 439-451.

35. H. J. Wacker (editor), Continuation methods, Academic Press, New York, NY, 1978.

36. A. C. Walker, A non-linear finite element analysis of shallow circular arches, Int. f. Solids and Structures 5, 1969, 97-107.

37. L. T. Watson, A globally convergent algorithm for computing fixed points of C^2-maps, Appl. Math. and Comp. 5, 1979, 297-311.

38. L. T. Watson and D. Fenner, Algorithm 555: Chow-Yorke algorithm for fixed points or zeros of C^2-maps, ACM Trans. Math. Software 6, 1980, 252-260.

39. J. W. Young, A. A. Schy and K. G. Johnson, Prediction of jump phenomena in aircraft maneuvers, including nonlinear aerodynamic effects, J. of Guidance and Control 1, 1978, 26-31.


DATE
FILMED
8